NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A MODULAR SIMULATION FRAMEWORK FOR ASSESSING SWARM SEARCH MODELS

by

Blake M. Wanier

September 2014

Thesis Advisor:	Timothy H. Chung
Second Reader:	James Eagle

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	9-26-2014	Master's Thesis 09-17-2012 to 09-26-2014

4. TITLE AND SUBTITLE

A MODULAR SIMULATION FRAMEWORK FOR ASSESSING SWARM SEARCH MODELS

5. FUNDING NUMBERS

6. AUTHOR(S)

Blake M. Wanier

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: N/A.

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The ability to utilize large numbers of unmanned systems as search agents allows the implementation of different search strategies that are not currently explored utilizing today's search decision support and analysis tools. This thesis develops a framework in MATLAB that allows the investigation of search strategies that utilize large numbers, or a swarm, of search agents. By implementing a modular design, multiple aspects of the search, such as tactics, searcher characteristics, and target characteristics, can easily be varied and analyzed. Utilizing JMP to perform statistical analysis, future design requirements can be refined in order to advise decision makers on possible alternatives and trade spaces for optimizing swarm search performance. Numerical studies demonstrate the ability to leverage the developed simulation and analysis framework to investigate three canonical swarm search models as benchmarks for future exploration of more sophisticated swarm search scenarios.

14. SUBJECT TERMS

Swarm Search, Search Theory, Modeling Framework, JMP, MATLAB, Analysis Tool, Modular, UAV, UXS, ASW

15. NUMBER OF PAGES 103

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A MODULAR SIMULATION FRAMEWORK FOR ASSESSING SWARM SEARCH MODELS

Blake M. Wanier
Lieutenant, United States Navy
B.S., United States Naval Academy, 2007

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 2014

Author: Blake M. Wanier

Approved by: Timothy H. Chung
Thesis Advisor

James Eagle
Second Reader

Robert Dell
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The ability to utilize large numbers of unmanned systems as search agents allows the implementation of different search strategies that are not currently explored utilizing today's search decision support and analysis tools. This thesis develops a framework in MATLAB that allows the investigation of search strategies that utilize large numbers, or a swarm, of search agents. By implementing a modular design, multiple aspects of the search, such as tactics, searcher characteristics, and target characteristics, can easily be varied and analyzed. Utilizing JMP to perform statistical analysis, future design requirements can be refined in order to advise decision makers on possible alternatives and trade spaces for optimizing swarm search performance. Numerical studies demonstrate the ability to leverage the developed simulation and analysis framework to investigate three canonical swarm search models as benchmarks for future exploration of more sophisticated swarm search scenarios.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

CDF	Cumulative Distribution Function
CDP	Cumulative Detection Probability
DOD	Department of Defense
DoE	design of experiments
GUI	Graphical User Interface
HVU	High Value Unit
MAD	Magnetic Anomaly Detector
MOE	Measure of Effectiveness
MOP	Measure of Performance
NOLH	Nearly Orthogonal Latin Hypercube
NPS	Naval Postgraduate School
OPLAN	Operation Plan
P_D	Probability of Detection
PACFLT	Pacific Fleet
SRU	Search and Rescue Unit
UAV	Unmanned Aerial Vehicle
USCG	United States Coast Guard
USPACOM	United States Pacific Command
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle

UXS	Unmanned System

Executive Summary

Information provides a distinct advantage in the battlespace. Knowing the strength and disposition of an opponent's forces allows the efficient allocation of one's own resources. Therefore, it is vital to possess not only the capability to gather information, but also the ability to utilize those information-gathering resources efficiently. The advent of unmanned systems and their introduction into the battlespace provide the capability to quickly gather large amounts of data [1]. For many such systems, their relatively small size, inexpensive cost, and the ability to save lives by keeping pilots out of dangerous situations make them ideal reconnaissance assets, especially in dangerous situations [2]. Unfortunately, limitations in command and control of unmanned systems as well as legacies from utilizing manned systems have prevented the use of large numbers of unmanned systems as efficiently as possible. Recent advances in technology and autonomous control allow for the coordination of large numbers of unmanned systems, often termed *swarms* [3]–[5]. For their future employment, it is necessary to develop new methods of gathering information and performing searches in light of these new capabilities [6]. The development of an analytic and simulation framework for designing, simulating, and assessing swarm search capabilities motivates this thesis, with the goal of providing a foundation for follow-on research.

First we define a simple scenario that guides the development of our model. We then develop a simulation framework in MATLAB in order to investigate the scenario. We utilize a modular approach, breaking the simulation down to its core components, and developing them in such a manner that any module can be isolated, replaced, and/or adapted to facilitate the investigation of different search patterns and strategies. Following the development of our simulation framework, we implement three search patterns, namely Agents Abreast, Agents in Column, and Random Walk, to address the search problems in the scenario, as well as demonstrate the flexibility of the framework.

In designing our experiment, we investigate the advantages and disadvantages of several well-known experimental designs. We implement a Nearly Orthogonal Latin Hypercube design for the generation of our design points in order to leverage its space-filling property with a relatively small number of design points [7]–[10]. We utilize this design to inspect

how different searcher characteristics affect search performance, including the number of searchers, the speed of the searchers, and the spacing between them. To analyze our data, we use the regression capabilities of JMP. In particular, stepwise and standard least squares linear regression on the Monte Carlo simulation show that nearly all of the inspected factors contribute to our model, but the number of searchers is the most significant for the Agents Abreast and Random Walk search patterns, and searcher spacing and velocity are the most important for the Agents in Column pattern. Though the presented scenario only analyzes a reduced number of agent characteristics and search patterns, the analysis demonstrates that the framework is capable of modeling different search patterns and providing results in a form ready for analysis.

The presented work also illustrates several potential operational applications of this framework, as well as aspects of the framework that can be improved in future research in the hopes of encouraging further analysis in this developing field of study.

List of References

[1] S. J. A. Edwards, *Swarming on the Battlefield*. Santa Monica, CA: Rand, 2000.

[2] R. O. Work and S. Brimley, "Preparing for war in the robotic age," Center for a New American Security, Washington, DC, Tech. Rep., 2014.

[3] D. J. Nowak, I. Price, and G. B. Lamont, "Self organized UAV swarm planning optimization for search and destroy using swarmfare simulation," in *Winter Simulation Conference*, Dayton, OH, 2007, pp. 1315–1323.

[4] L. Barnes, M. M.-A. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Mediterranean Conference on Control Automation*, Athens, Greece, June 2007, pp. 1–8.

[5] J. A. Sauter, R. S. Mathews, K. Neuharth, J. S. Robinson, J. Moody, and S. Riddle, "Demonstration of swarming control of unmanned ground and air systems in surveillance and infrastructure protection," in *IEEE Conference on Technologies for Homeland Security*, Boston, MA, 2009, pp. 51–58.

[6] J. J. Corner, "Swarming reconnaissance using unmanned aerial vehicles in a parallel discrete event simulation," M.S. Thesis, Air Force Institute of Technology Wright Patterson AFB OH School of Engineering and Management, 2004.

[7] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "A users guide to the brave new world of designing simulation experiments," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 263–289, Aug. 2005.

[8] S. Sanchez and H. Wan, "Work smarter, not harder: A tutorial on designing and conducting simulation experiments," in *Winter Simulation Conference*, Berlin, Germany, 2012, pp. 47–57.

[9] S. M. Sanchez, T. W. Lucas, P. J. Sanchez, C. J. Nannini, and H. Wan, "Designs for Large-Scale Simulation Experiments, with Applications to Defense and Homeland Security," in *Design and Analysis of Experiments, Special Designs and Applications*, K. Hinkelmann and O. Kempthorne, Eds. Hoboken, New Jersey: John Wiley & Sons, 2012, vol. 3, no. 1986, ch. 12, pp. 413–442.

[10] S. M. Sanchez and T. W. Lucas. SEED Center for data farming. (2014, Jul. 11). [Online]. Available: http://harvest.nps.edu/

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank my advisor, Timothy Chung, for always encouraging me to learn more about this topic and pushing me to work harder and achieve results well beyond my expectations. His support and enthusiasm at all times of the day and night broadened my horizons and made this an enjoyable experience. Thank you for being a mentor, an inspiration, and a sounding board to work through my ideas.

I would also like to thank Professor Eagle, for helping me realize the potential impact this work can have. Thank you for helping ground my abstract ideas into real world scenarios. Your expertise in the realm of operations research helped shape my path, guiding me always onwards towards my goals.

I would like to thank my wife, who even after the stress of this work still decided to marry me. Your love and support buoyed my spirits throughout the entirety of this colossal effort. Without your endurance and patience, I would not be where or who I am today.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Background and Motivation

1.1.1 Changing Undersea Search Environment

Military capabilities change and advance as new technologies are developed and integrated into military platforms. While these advances greatly improve our own capabilities, they also provide new areas for the enemy to exploit. Over the past several decades, the capabilities of undersea platforms have greatly improved with smaller acoustic signals making them harder to find and track by conventional acoustic methods [1], [2]. Though the U.S. and allied military forces may compensate for this by employing other sensors, such as magnetic anomaly detectors (MAD) in the search for underwater vehicles. However, limitations in the capabilities of these sensors need to be overcome before they are completely effective. By exploring new tactics to search for and track underwater vehicles, this thesis seeks to find more efficient and effective means to address these emergent threats.

Advances in technology are not limited solely to the undersea environment. Recent significant advances in Unmanned Systems (UXSs) create an immense new field to explore, providing unique advantages in many warfare areas. One advantage of UXSs is the ability to employ them in large numbers, such as in swarms [3]. While each individual platform provides capabilities less than those of current warships or aircraft, when large numbers are considered in aggregate, they can quickly surpass the abilities of a single asset and potentially at significantly less cost [4]. Furthermore, the implementation of UXSs nearly eliminates the risk to human lives and to the conventional platforms they replace, providing an immeasurable benefit in reducing human losses. By using autonomous swarms with the proper tactics, it might be possible to significantly improve search and track capabilities in anti-submarine warfare missions.

The Peoples Republic of China is rapidly becoming a world power and is placing an increased emphasis on maritime control of the Western Pacific. In order to accomplish these goals, China has increased her submarine inventory, and her neighboring countries have

responded in a similar manner [1]. Meanwhile, the U.S. submarine inventory continues to decrease. While an increase in the number of UXS provides some compensation for the decreasing inventory, current capabilities and tactics do not allow UXS to replace submarines. Creating a method to investigate the operational utility of UXS is vital to ensure the development of these systems into an effective tool.

The tactical potential of swarms has yet to be fully investigated. Also the communications and computational requirements for operating a swarm are difficult to estimate [5]. Unfortunately, most current search strategies and analytic tools are designed around the employment of a single search agent, or perhaps a few, and are not suited for a large number of searchers [6].

1.1.2 Previous Work

Previous areas of research have often focused on search methods derived from nature [7]. One example of such bio-inspired models is based on the method ants use for foraging. By using a limited number of indicators, often referred to as "digital pheromones," large numbers of unintelligent agents are able to coordinate and act in an intelligent manner [5], [8], [9]. Engelbrecht explains how even without a central controller, workers know when to reassign themselves to different tasks [10]. This type of search coordination using indirect communication, or stigmergy, could prove very effective where there is limited communication bandwidth, and where the searchers carry out most of the computing. Dasgupta [11] illustrates why automatic target recognition by Unmanned Aerial Vehicles (UAVs) is more challenging than the stigmergic behavior of ants. There exists an absence of physical medium for UAVs to leave trails for other agents. Furthermore, Dasgupta reiterates that unlike for ants, the target is often mobile. As such, the pheromone trail cannot be reused for all future travel to the target, unlike ants searching for food. Lastly, the random movement of ants is not suited for UAVs due to flight path conflicts, and other search paths with few turns might be more effective. It is important to extract what we can from these nature-based models while recognizing the limitations that UXSs must work within.

Swarm simulation research done at the Air Force Institute of Technology examined the interactions between agents. The main goal of its research is to investigate swarm behavior which emerges autonomously. Communication between agents in this work was

kept to a minimum, limited to contact only with their closest neighbor and limiting the information passed to only their current position and vector. The agents determine their behavior primarily through implicit communication, very similar to the ant pheromones seen in nature [12]. Furthermore, Banks and Vincent [13] explored the trade-offs between deterministic search and various stochastic search models. Their research highlighted the importance of the target distribution in determining the effectiveness of search patterns. If *a priori* information existed indicating target clustering, then many of the bi-phase strategies proposed by the authors proved more effective, as they switched to more intensive search patterns when a target was discovered. When no *a priori* information was available, or little clustering, then deterministic patterns proved to be more effective [13].

The ability to control robot swarm formations is in development, and some solutions appear to be feasible. Barnes *et al.* [14] investigated UXSs control using potential fields and proposed a model which scales to various swarm sizes and models. Their method utilizes potential functions together with limiting functions to successfully control robot swarm formations. It also supports scalability, multiple formations, and heterogeneous swarms, while remaining computationally inexpensive. This provides a foundation for the formations that will be utilized in swarm search models.

1.2 Research Objectives

It is the goal of this research to provide an analytic and simulation framework for designing, simulating, and assessing swarm search capabilities. This framework would act as a sandbox capable of exploring numerous distinct swarm search scenarios including different search patterns, different sensors and detection curves, different agent and target characteristics and behaviors. In doing so, we hope to help shape emerging technology and agent designs, help develop and investigate search strategies that will aid operators in the field, and help provide decision makers with the necessary tools to make informed decisions on resource allocation and search doctrine.

1.2.1 Research Questions

When developing this framework, there are many problems and questions that we can address. Most of these questions include challenges that makes for complex implementation.

The following provides a list of potential questions research in this area can address, and some of the technical challenges associated with them.

1. What search patterns provide the most effective means of finding a target? Are there significant advantages to using one over another?
   Technical Challenges:
   (a) Coordinating search between many agents
   (b) Modeling learning for an intelligent adversary
2. Which Measures of Effectiveness (MOE), i.e., time to first detection, probability of detection, etc., gain the most advantage from employing many UAVs?
   Technical Challenges:
   (a) Providing the decision maker with the information necessary to make an informed decision on the numbers and capabilities of UXSs
3. Is it better to partition the search area or coordinate search for multiple looks?
   Technical Challenges:
   (a) Determining optimal allocation of limited resources
4. Does leveraging multiple types of sensor provide a distinct advantage? Does one type of sensor distinctly outclass others?
   Technical Challenges:
   (a) Acquiring accurate models of lateral range curves for multiple sensors
   (b) Computational requirements of using multiple sensors in a single model
5. What advantage does each additional UXS provide in prosecuting underwater targets, from detecting and identifying to tracking and engaging?
   Technical Challenges:
   (a) Developing a model capable of analyzing a wide range of factors over multiple levels
6. Is a significant advantage gained from employing multiple UXSs across multiple domains? Does one type of vehicle greatly trump the others?
   Technical Challenges:
   (a) Developing heterogeneous model to account for different searcher and sensor performances
7. What level of communication is required between search vehicles? Does each vehicle need to be in constant contact with all of the other vehicles? Does a hub-and-spoke

or closest neighbors scheme provide enough efficiency?

Technical Challenges:

(a) Developing limited peer-to-peer and mesh communication schemes and algorithms to model them

8. Can we develop a scalable model capable of being implemented for any number of UXSs?

   Technical Challenges:

   (a) Developing algorithms for real time partitions of the search area based on the number of UXSs
   (b) Developing algorithms for communications between agents
   (c) Computational requirements of modeling many UXSs operating independently

While each of these questions provides a direction for further research, this thesis focuses on questions one, two, three, and eight. By developing a flexible framework capable of providing statistical analysis on a wide range of search patterns and configurations, this thesis provides a tool capable of addressing the technical challenges enabling the analysis necessary to answer those questions.

1.2.2 Benefits of the Study

The main contribution of this research is a simulation framework based on MATLAB capable of running search simulations incorporating large numbers of searchers. This framework incorporates numerous aspects of swarm search, including agent and target speed, search area, and the number of searchers, implemented as configurable modules, such as searcher, target, and detection modules. This provides the user with relevant data on what factors impact the search, in order for them to be able to make informed decisions. This framework includes the ability to perform further analysis on many-agent searches in a wide variety of applications, including against multiple and/or moving targets. This tool allows the development and refinement of search strategies for varied situations, in addition to providing a means of sensitivity analysis to assess the benefits gained by changing aspects of the search, such as the number of searchers. We seek to provide the decision maker with a quantitative assessment on the advantages of employing additional resources, as well as what factors provide the greatest benefits to assist in future search platform design and development. Such analysis supports operational stakeholders that perform

searches on regular basis, by reducing search time, reducing resource requirements, and increasing probability of detection. Furthermore, the framework is not restricted solely to searching for underwater assets. By utilizing a different `target` module, the United States Coast Guard could simulate search and rescue operations for personnel lost at sea, or the `searcher` module could be altered to guide the deployment of multiple UAVs searching for submarines forward of a strike group. The MATLAB simulations proposed in this thesis may greatly benefit many groups while expanding the potential of swarm search and detection.

1.3 Scope, Limitations, Assumptions

1.3.1 Scope of Thesis

This thesis addresses the development of a scalable simulation and analysis model for swarm searches. We approach this task by developing a modular framework, providing the capability of quickly changing any part of the search model without affecting the performance of the rest of the model. We also demonstrate the statistical analysis necessary to determine how factors impact our model. In order to accomplish this, we demonstrate our framework through the development and analysis of three case studies involving the analysis of searcher performance and their effectiveness relative to each other. Previous works demonstrate that the capabilities exist to perform searches utilizing large numbers of agents [15]–[17]. We seek to utilize this capability to develop a method to analyze the relative importance of the factors that contribute to an effective swarm search.

1.3.2 Limitations

- This work does not attempt to determine the most effective search strategies for swarms. Rather, this thesis provides a framework where different models may be designed and assessed.
- This work does not experiment with different sensors models. Such analysis can be investigated through the use of the `detection` module which simulates the sensor and detection process.
- This work does not investigate different experimental designs. Rather, we utilize a Nearly Orthogonal Latin Hypercube (NOLH) design to investigate the effectiveness of our framework and leave exploration of other statistical designs to future work.

1.3.3 Assumptions

A number of assumptions further limit the scope of this thesis:

- Space and time are discrete.
- UAVs possess identical capabilities and characteristics.
- UAVs possess perfect knowledge of their own locations.
- UAVs can communicate without error or delay.
- UAVs do not need to perform evasive or counter-detection maneuvers.
- There are no environmental effects that affect flight paths.

1.4 Thesis Organization

This chapter introduced the problem, including the motivation and relevant background of utilizing UAV swarms in the search for small or hard to detect targets. In Chapter 2, we describe the scenario we used for our research in addition to developing the simulation model. In Chapter 3, we describe the experiment that we conduct and explain the factors considered in our model. In Chapter 4, we present the results of the simulated case studies demonstrating the capabilities of our framework. In Chapter 5, we present the conclusions of our research and provide recommendations for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Model Formulations

This chapter contains a brief introduction to search theory followed by a description of the swarm search scenario; it concludes with a detailed outline of the simulation and analytic framework explored in this thesis. The framework is flexible enough to model numerous different scenarios, while maintaining the complexity required to garner useful insights from our simulations. We capture the insights that a stochastic model provides by utilizing the Monte Carlo method in generating replications.

2.1 Background

Throughout the twentieth century, search theory has provided valuable tools to decision makers. The development of unmanned systems capable of acting both as the hunter and the hunted has motivated many search problems. Search, at its heart, is information gathering and relies primarily on two components: uncertainty in the location of the target and uncertainty in the information provided by sensors [18]. In the ideal situation, the searcher knows the exact location of the target and possesses an ideal sensor. The definite-range law, or cookie-cutter, sensor is an ideal sensor, always detecting the target when within range. Introducing location uncertainty but maintaining the ideal sensor provides the basis of our initial search problems. These coverage problems often seek to maximize one of two objectives: the time to first detection or time to complete coverage. It is in this context that the concept of an exhaustive or ideal search becomes clear. A search is considered to be ideal if the searcher utilizes an ideal sensor, and does not overlap its coverage nor expend search effort outside of the search area [6]. In most real-world situations, it makes sense to picture this search pattern as a lawnmower cutting grass row by row as generally it is not possible to jump over large areas instantaneously. As Washburn [6] discusses, assuming an ideal search and sensor results in an upper bound on the performance of realistic area searches. In Equation 2.1, v is the velocity of the searcher, w is the searcher's sweep width, A is the total search area, t is time, and $F_{\text{ideal}}(t)$ is coverage ratio, namely the percentage of the area searched for a single searcher conducting an ideal search as a function of time.

$$F_{\text{ideal}}(t) = \frac{vw}{A}t \tag{2.1}$$

We replace $\frac{vw}{A}$ with γ, also known as the *coverage rate*, to simplify the expression. In cases where repeated coverage of the search area must be performed in order to detect the target, the coverage factor may exceed one [19], and we define t^* as the time to perform one complete coverage of the search area. Furthermore, in the case where we utilize an ideal search and sensor, which we assume for the remainder of this section for illustrative purposes, the coverage ratio bounded by 1 is also the cumulative detection probability as seen in Equation 2.2 and Figure 2.1.

$$P_{D,\text{ideal}}(t) = \min\left(F(t), 1\right) = \begin{cases} \gamma t, & t \leq t^* \\ 1, & t > t^* \end{cases} \tag{2.2}$$

By combining the search rate of multiple searchers, we are able to expand the above expressions to incorporate these M searchers as seen in Equation 2.3 and illustrated in Figure 2.2, where γ_i is the coverage rate of agent i.

$$F_{\text{ideal}}(t) = \begin{cases} \sum_{i=1}^{M} \gamma_i t, & t \leq t^* \\ 1, & t > t^* \end{cases} \tag{2.3}$$

Conversely, if the agent randomly moves through the search area, significant overlap in coverage may occur. During the initial stage of the search, there is very little area that has been covered, so there is only a small likelihood of overlap of the coverage, but as more of the search area is covered, the random search ends up overlapping more and more of the search area [19]. This wasted search effort makes random search considerably slower at ensuring that any specified fraction of the search area is covered. Random search provides us with a lower bound on intelligent searches, as represented by Equation 2.4.

$$F_{\text{rand}}(t) = 1 - e^{-\gamma t} \tag{2.4}$$

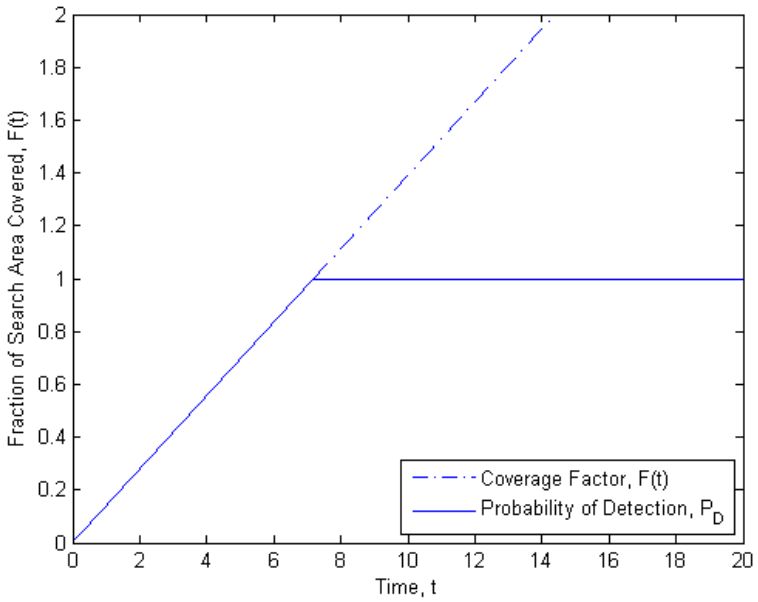


Figure 2.1: The linear increase of the coverage factor and Probability of Detection as time progresses a single searcher. Whereas the coverage factor linearly increases past one indicating multiple coverages of the search area, the probability of detection represents the probability of having detected the target, and is bounded by one.

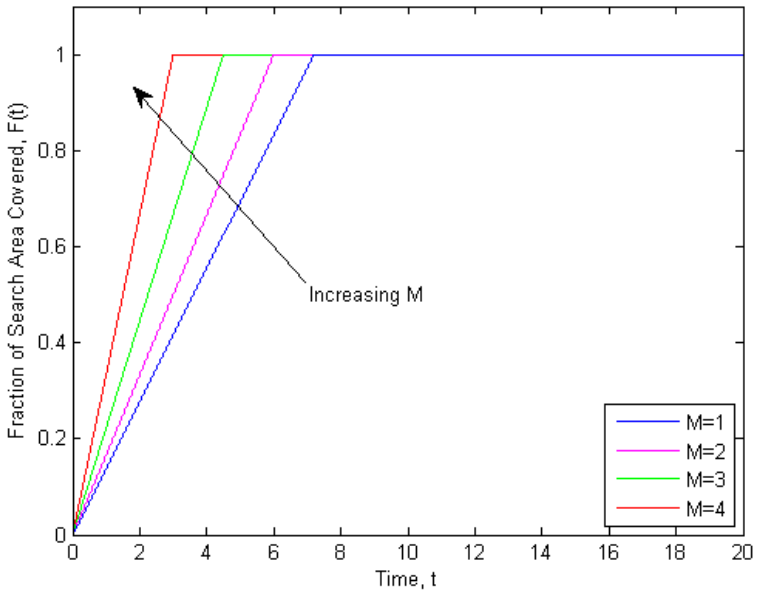


Figure 2.2: The linear increase of the coverage factor as time progresses for one, two, three, and four agents. This type of ideal search provides the upper bound on coverage rate.

It is possible to do worse than random search, but this involves a deliberate attempt to achieve a low detection probability. Similar to the ideal search, we can expand the equation to incorporate multiple agents by combining the efforts of their search as seen in Equation 2.5. As expected the coverage factor increases more slowly as time progresses than that for the ideal search as seen in Figures 2.3 and 2.4 [6].

$$F_{\text{rand}}(t) = 1 - e^{-\sum_{i=1}^{M} \gamma_i t} \tag{2.5}$$

The next step in search modeling is to characterize the detection model, that is, identify the uncertainty in the sensor. In other words, just because the sensor passes over the target does not mean that the target is necessarily detected, and similarly, just because the sensor detects a target does not mean that the target is actually at that location [6]. While the aforementioned models assumed a perfect sensor, this extension of imperfect detections provides significantly more variability to the model and makes it necessary to determine where and for how long to look (in continuous cases). Since we are limited by the amount of resources available, optimized allocation becomes very important. The development of unmanned systems potentially allows for large numbers of agents to be employed simultaneously in searches, leading to the investigation of similar searches done in nature. As a possible avenue for further exploration, one can study foraging theory, which attempts to mimic the hunting methods of various species [13], [20]. A potential application to many unmanned systems may also be seen by ant behaviors where each agent relies on "pheromones" in order to determine its own role in the search as well as to communicate with other agents participating in the search [21].

For our research, we build a MATLAB framework capable of investigating these different models. By implementing a modular design, we enable the rapid exploration of a wide range of models and parameters. This structure decomposes the model into modular sub-functions, with key aspects of the model passed to these sub-functions. This allows restructuring of the model, either through changes to the imported data, or through modification of one or more of the sub-functions to alter the behavior of one or more aspects of the model. We then explore the robustness of our model by implementing several different

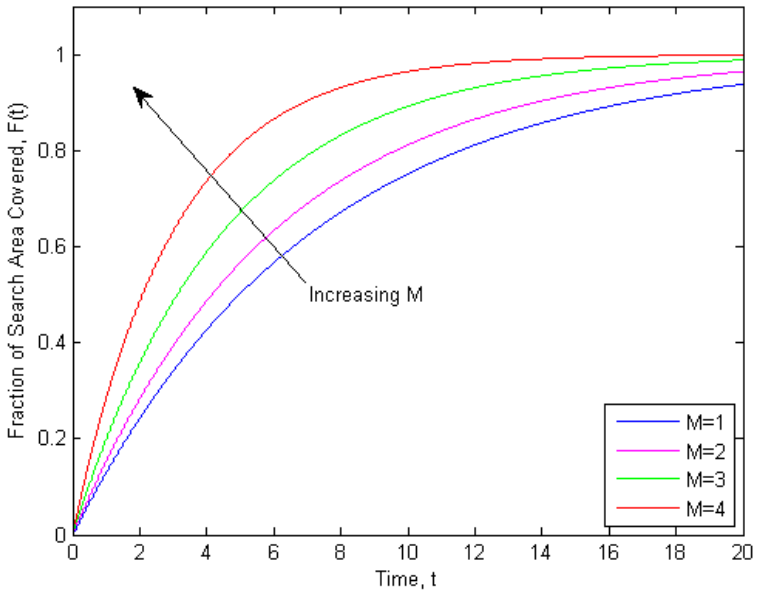


Figure 2.3: This figure demonstrates the sub linear increase of the coverage factor as time progresses for one, two, three, and four agents. This type of search provides the lower bound on coverage performance.

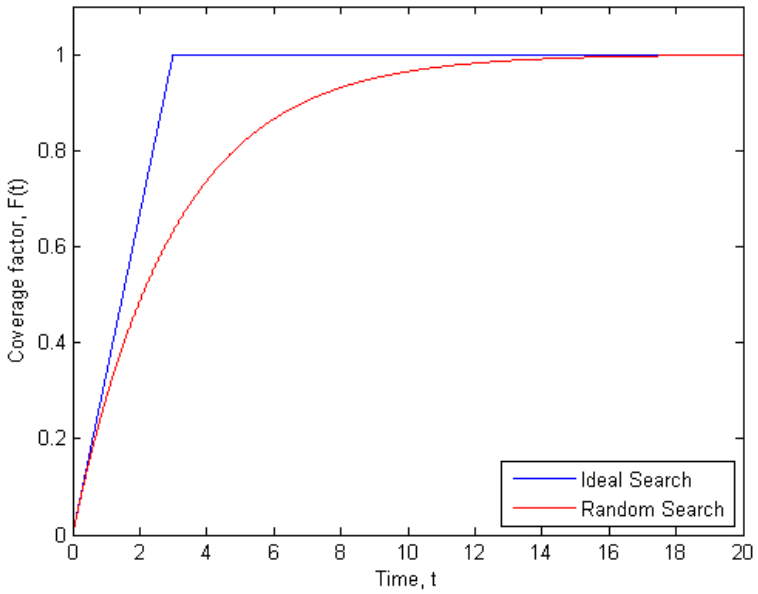


Figure 2.4: The difference between the upper bound of the ideal search with the lower bound generated by the random search.

swarm search model types, and investigate the ability to statistically analyze our resulting data.

2.2 Motivating Scenario

We present a hypothetical scenario from OA4602: Joint Campaign Analysis, Winter 2014, that provides a context in which a future capability of swarm unmanned systems to conduct search missions may become a necessity [22]–[28].

2.2.1 Background

In the years leading up to 2025 there has been a continuing growth of economic competition and demand for energy resources between China and Japan. China demonstrates an increasing mercantilist behavior, and territorial expansionism in their pursuit of natural resources. as well as strengthening ties to other Asian countries to include Taiwan, North Korea, and Russia. Japan responds to these actions by developing a larger, more capable defensive force that is less reliant on the United States, as well as strengthening ties to her U.S. ally through increased number of exercises and multiple joint bases. The United States has also increased development of its logistic infrastructure in the western Pacific, to include having secured additional basing rights. Furthermore, ASEAN is researching new methods to counter Chinese aggression in the area.

2.2.2 Scenario

As large portions of shipments of natural gas from Indonesia and Malaysia are redirected to China to support its growing domestic consumption, Japan's discovery of large natural gas and oil deposits in the East China Sea comes at a critical time for the country. China immediately claims sovereignty over these natural resources and threatens the use of force against Japan if these claims are not respected. The situation continues to escalate on both sides, with Japan deploying naval forces to protect their ships, while China deploys its subsea and air assets in shows of force, and warns the United States and Japan of their intention to quarantine Okinawa if there continues to be infringements into China's claimed economic exclusion zone resources. In addition, the United States, Philippines, Indonesia, Japan, Singapore, and Vietnam all register their protests against Chinese "safety inspections" of tankers traveling through the South China Sea, which is a clear infringement of freedom of the high seas.

2.2.3 Objective

United States Pacific Command (USPACOM) has been tasked with responding to the loss of freedom of navigation, and if China begins a full denial campaign, establishing sea and air control as well as defending land bases in the region. In efforts to accomplish this goal, USPACOM tasked Pacific Fleet (PACFLT) to develop an Operation Plan (OPLAN) to ensure freedom of navigation, monitor and counter potential deployment of Chinas South Sea submarine fleet into the East China Sea, and engage and counter Chinese surface forces if hostilities arise.

China's submarines, possessing one of the deadliest anti-ship missiles, pose a significant threat to the United States and Japanese surface fleets. As such, it is vitally important to counter the Chinese submarine fleet in the East China Sea. Currently the United States possesses a strong advantage in the undersea realm, and with the forecasted acquisition schedule for all parties involved, it is imperative that the United States retains that advantage. Furthermore, the greatest challenge facing the Fleet is the time it takes to hunt the Chinese submarines, given the immense search area of the East China Sea. Any enhanced ability to localize the Chinese submarines greatly reduces the time required to counter their forces.

Numerous developments in the command and control of unmanned systems creates the ability to utilize large numbers of UAVs in order to search for enemy submarines with swarms. Furthermore, advancements allow miniaturized detection technologies to be mounted on smaller vehicles, making it possible for many of these systems to deploy from smaller ships, rather basing them on aircraft carriers. It is necessary to determine if it is more effective to leverage these capabilities by employing large numbers of smaller, but nominally less capable, unmanned systems to search for their targets, rather than employment of a few larger, but potentially more capable search units.

The goal of the presented study in the context of this scenario is to provide the tools to USPACOM to determine effective submarine prosecution tactics with a large number of search assets, as well as the necessary capabilities that future UAVs must possess to effectively carry out their mission. In this theme, we seek to provide a framework to allow the testing of search tactics and the effectiveness of various UAVs platform and sensor

characteristics in a variety of environments and roles. USPACOM's interests include, but are not limited to, how the following parameters impact the time to first detection of submarines: the number of search agents they should include in their searches; the dimensions of their partitioned search areas; the speed of the search agents; the search tactics and/or search patterns employed; and the enemy's behavior and capabilities. For generality and to avoid classification issues, we use generalized searchers and sensor models for the presented analysis. In this manner, we build the framework to address many situations without limiting ourselves to a particular platform and sensor type, although specific insights may be obtained by use of actual specifications of systems of interest.

2.3 Searcher Descriptions and Parameters

In order to develop a robust framework, we leave the characteristics of the search model, including searcher speed, detection models, and number of searchers, as variable parameters. We analyze our models assuming a specified large number of search agents, each with a constant search speed and using a triangular detection curve for their probability of detection. We choose to utilize all searchers with the same speed to represent a search with multiple agents of the same type performing a coordinated search. Similarly, the triangular detection curve is a simplified yet representative model of many sensors that exhibit better detection performance the closer the target is to the sensor, such as radar and sonar. In this generic model, the probability of target detection decreases linearly from 0.2 to 0 as the horizontal range increases from 0 to 20 NM, as illustrated in Figure 2.5. Furthermore, the searcher performs one detection attempt per timestep, which is set to one minute for this study.

Similarly, we assume that the single target we model is generic in order to demonstrate the adaptability of our framework. The target possesses no knowledge of and is unable to detect the agents searching for it, but assumes that it could be hunted. Therefore the target behaves in an erratic manner, randomly changing directions but maintaining a constant speed throughout each scenario. Future work can investigate alternate models of target behaviors, to include reactive strategies that might seek to evade the search.

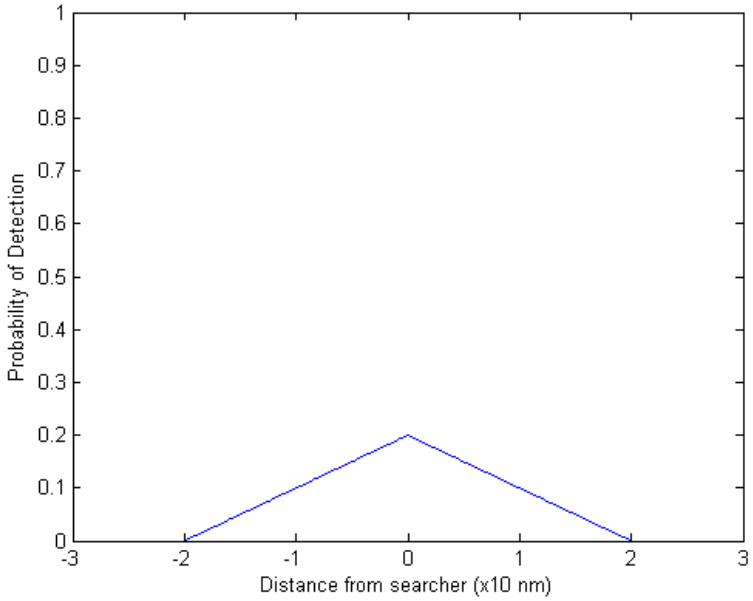


Figure 2.5: A graphical representation of the sensor we utilize in our models. We model our sensors by a triangular curve with the highest chance of detection directly below the agent, decreasing linearly to 0 at 20 NM away.

2.4 Model Assumptions

It is generally impossible or impractical to model every aspect of a system. Not only are there too many variables that might affect the outcome, that complex of a model may provide little useful data to a decision maker. Therefore, we make assumptions to simplify the model in order to hopefully provide useful insights for the user.

- Static Search Area: We restrict the location of the target within a static search area, and the search area does not expand nor change shape as the scenario progresses. We utilize a rectangular search area to simplify the movement of the agents and minimize overlap of search effort.
- Triangular Detection Curve: Rather than implementing a detection curve for a specific searcher and sensor type, we utilize a generic and fixed detection curve as an initial detection model which we can change to determine how different sensors perform in the search environment.

- Model analog detectors as discrete elements: Since the model requires computational implementation and actions occur during discrete timesteps, it is not possible to easily model continuous elements. We therefore model analog and continuous processes, such as movement of agents and the occurrence of independent detection opportunities, to happen once each timestep, which is set to one minute for this study.
- Weather and sea-state do not significantly impact the search model: We assume that weather and sea-state conditions do not effect factors such as sensor performance, velocity, and spacing; by doing so, we reduce unnecessary complexity in the search model that might confound additional factors.

2.5 Development of Search Model

2.5.1 Search Patterns

In our analysis of the scenario, we begin by investigating three different search patterns. In the first search pattern we utilize multiple agents searching abreast in a lawnmower pattern as illustrated in Figure 2.6. In this model, all of the agents start in the lower left corner, maintaining formation to the starboard side of the guide. The guide acts as the focal point of the searchers' positions in the pattern, allowing the searchers to maintain the pattern by aligning to the starboard side of the guide. This greatly increases the effective sweep width of the sensor, allowing for a much faster coverage of the search area. The agents move in a typical lawnmower pattern, utilizing top-to-bottom sweeps of the search area, wheeling when reaching the top or bottom of the search area so relative position from the guide remains unchanged. After a complete sweep of the search area, the searchers reverse direction and begin the next sweep from their final position vice resetting to the bottom left corner of the search box.

In the second search pattern, we once again utilize multiple agents searching in a lawnmower pattern but rather than agents moving abreast, they form a column with each agent following the agent directly in front of them as seen in Figure 2.7. This causes the effectiveness of the sensor to increase, as each agent covers the same area providing multiple possible detections for the target on each pass. While this approach covers the area more slowly than the first pattern, it provides a much higher probability of detection on each pass. The general shape of the lawnmower search remains unchanged, starting at the bottom left

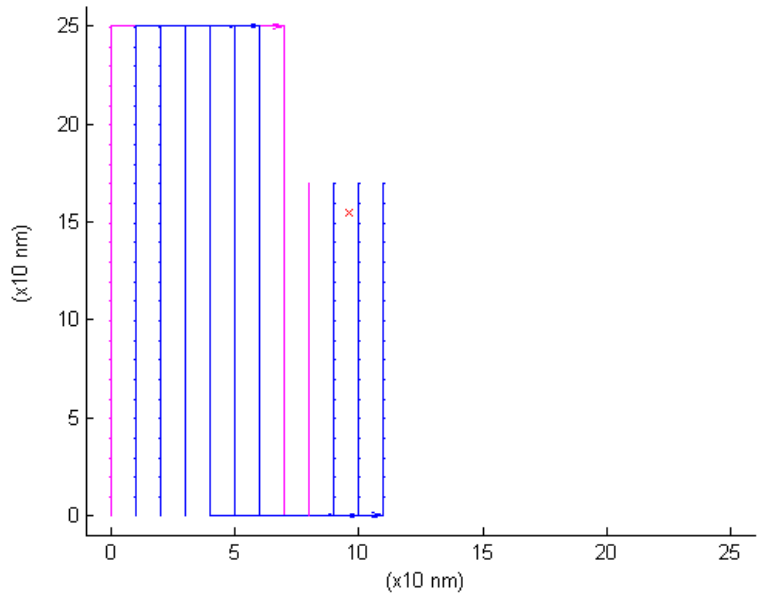


Figure 2.6: Agents Abreast search pattern with four agents spaced one unit apart. The guide agent, which the other agents use to maintain their position in the search pattern is highlighted in magenta; the target is denoted as the red $\times$.

corner of the search area and moving to the right. When the sweep is complete, the agents begin their next sweep from their last position.

In the final canonical search pattern explored in this thesis, we utilize multiple agents moving in a random fashion to search for the target as illustrated in Figure 2.8. Once again we start all searchers in the bottom left corner aligned abreast with their initial direction being towards the center of the search box. Each agent moves at a constant speed changing direction within a limited bearing from its current heading. When a searcher reaches one of the boundaries of the box, it changes direction towards the center of the search area before continuing its random movement. While this pattern is not as efficient in ensuring that the entire search area is covered, such a pattern makes it more difficult to predict where the search agents will be possibly providing a tactical advantage in hostile environments.

2.5.2 Measures of Effectiveness

For our framework, we are interested in assessing the effectiveness of the given search approach. As such, we center our model around generation of quantitative measures of

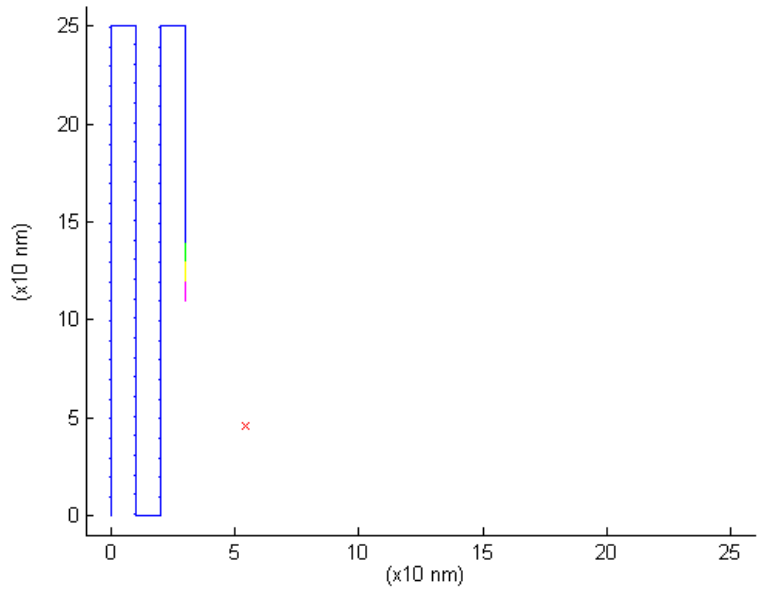


Figure 2.7: Agents in Column search pattern with four agents spaced one unit apart. The guide agent is magenta; the trailing agents are yellow, green, and blue, respectively. The target is the red $\times$.

performance that will help us to ascertain the efficacy of our search. Measures of performance represent the execution of specific aspects of the model; however, one must often manipulate or combine the measures of performance to generate relevant measures of effectiveness. It is important to carefully determine our measures of effectiveness, as often times they can be misleading [29]. Figure 2.9 demonstrates how we can utilize a fitted Cumulative Distribution Function (CDF) to measure the effectiveness of different search configurations. Analysis of the variance for the design points lets us concentrate our focus on the expected time to first detection as this provides us with a popular measure of effectiveness in search theory literature.

2.6 Framework Development

In this section we expand on the MATLAB software implementation we develop to support the proposed framework. Figure 2.10 illustrates a block diagram of the component software programs forming the structure of our framework. Each block represents a program or module with each level contained within its parent function but operating independently.

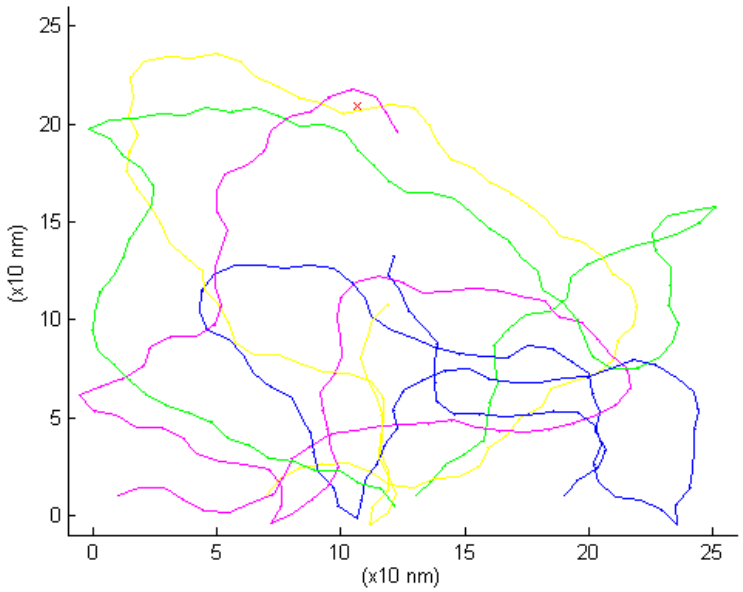


Figure 2.8: Random Walk search pattern with four agents spaced six units apart. There is no guide agent. The target is the red $\times$.

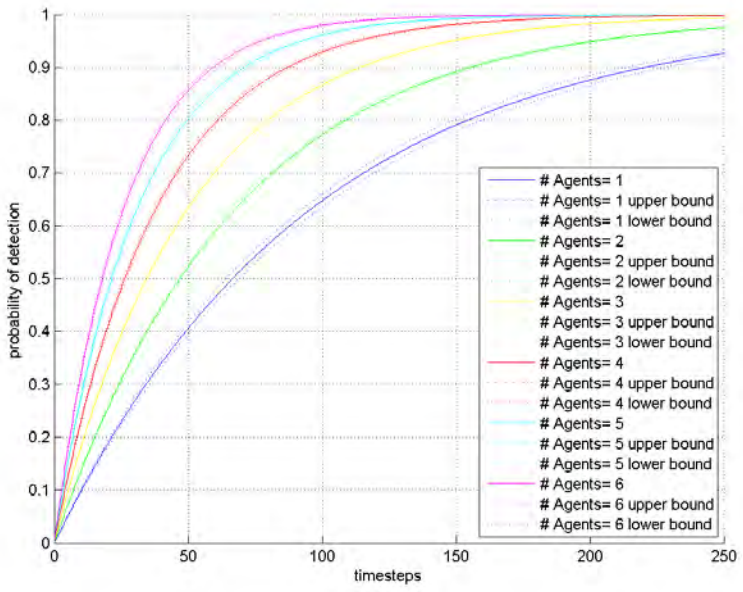


Figure 2.9: CDF changes as we increase the number of agents utilized during a random search. As expected, the performance improves as more agents are utilized, moving the curve up and to the left. This indicates that we have a higher probability of finding the target faster the more agents we use.

The arrows between blocks indicate the flow of model characteristics from the `main` function to the sub-functions as well as the results from searches to the `main` function.

2.6.1 Program Integration

The foundation of the framework is the `main.m` function. This function works to link the experimental design to the search model and the search model to the statistical analysis. This function begins by importing initial conditions such as the number of agents, the speed of the agents, and the size of the search area from a comma-separated value (.csv) file containing supporting data. This approach allows for significant flexibility in the design of any search model, since the design of the software is not intended to limit the number of factors investigated. For example, if the analysis of interest in a given search model requires varying only the number of searchers and number of targets, then only those data are necessary as inputs in the .csv data file. Similarly, if we change a module and thus need to include a factor (e.g., the velocity of the searcher), we can add that to our design changing the input data .csv to include that factor, but do not need to change any other part of the framework.

The `main` function also controls the Monte Carlo process by designating the design point for which we are currently generating replications. This is accomplished by fixing the design point, then passing the design point characteristics to the `search` function to generate the replication. After each replication, the `search` function passes the output response variables back to the `main` function for collating. When the requisite number of replications are complete, the `main` function saves the results both as a MATLAB-readable file serving as a backup and as a .csv file for later analysis. The MATLAB code we use to generate these replications and collate response results can be seen in Codeblock 2.1. After the main function saves the data from the design point, it begins on the next design point, appending the data to the previous files.

2.6.2 Replication Generation

The `search` module coordinates the individual replications. The `search` module utilizes the information in the design point to establish the search area by initializing a data structure to contain searcher and target data. It calls sub-functions to populate these structures with

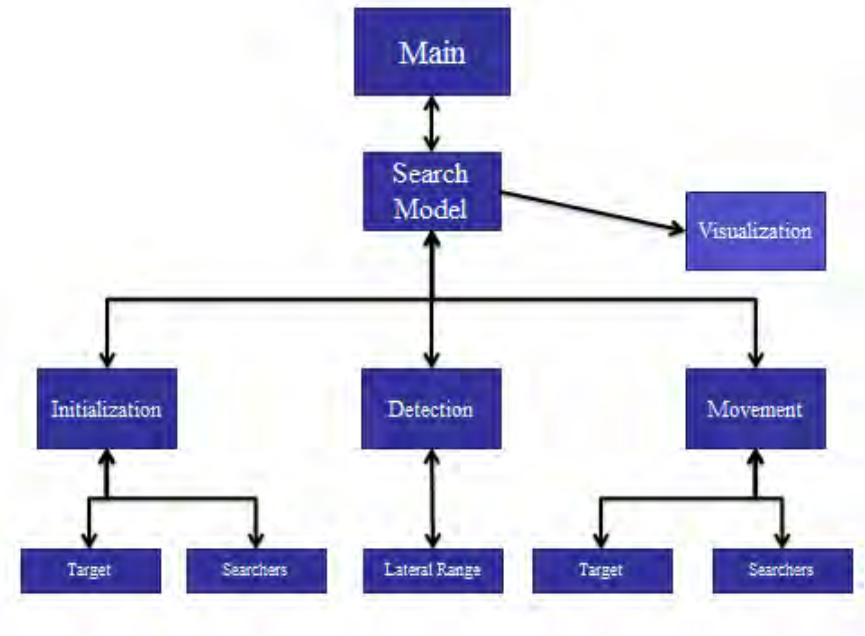


Figure 2.10: Block diagram of the framework developed. Each block represents a module or sub program with the arrows indicating the direction of information flow between the modules.

```
    dp_char=sim_char(j,:);

    % The following loop utilizes parallel threads to perform the search
    % function.  It passes onto the function the characteristcs of the
    % design point, as well as identifying which model to use, and whether
    % to display graphs.
    parfor i=1:runs
        [timestep,found,pass]=search(dp_char, modeltype, dispyn);
        % The following lines collect data to be saved to CSV and .mat file
        % so that further analysis or replication can be applied.
        jmpdata(k+i,:)=[dp_char,j,i,timestep];
```

Codeblock 2.1: The `main.m` function leverages multi-core processors to process multiple replications at a time, greatly reducing the runtime of the model.

initial position and movement data for the searchers and targets. Notice in Codeblock 2.2 searcher and target position as well as movement data are consolidated into a single data structure. This simplifies passing data to other sub-functions; as long as the data do not change in the sub-function they are not updated in the functional memory, thus saving space, time, and memory [30], [31]. Furthermore, these structures are not limited to just position and movement data, but can also include information such as current mission or behavior. The `search` function implements the search process by utilizing sub-functions to

determine detections, ascertain change in position of agents, and provide a visual representation of the search. The `search` function itself calculates the total number of detections, iterates the timestep, updates the target location, and returns to the `main` function when and if the target is detected. The `search` function controls and coordinates the different aspects of the search, such as the initial agent positions, agent movement, and target detections, returning to the `main` function the time that the target was found, the total number of agents that found the target, and the total number of missed and successful detections.

```
% Initialize searcher position based off the model type used.
[ag_data{1,1},ag_data{2,1},rtlc]=searcher_initial_posit(dp_char, modeltype);

% Initialize target position.
[ag_data{1,2},ag_data{2,2}]=target_initial_posit(dp_char);
```

Codeblock 2.2: Initialization of the searcher and target locations area is called from the `search.m` function, and handled by the respective sub-functions.

The sub-functions that control the initial position share the same role for both searchers and target but vary slightly in their execution. Both functions utilize the design point characteristics to determine initial position inside of the search area and their initial heading. Whereas the `target position` sub-function generates a random position for the target, the `searcher position` sub-function determines their starting positions based on their spacing from the bottom left corner of the search area. Both of these modules can be easily altered or replaced in order to capture a different aspect of initial target or searcher distribution. For example, if we have intelligence indicating the target's initial position is biased towards one side of the search area, we can incorporate that into the `target position` sub-function. Similarly, if we have agents arriving from multiple locations, we can initialize their starting locations in multiple spots around the search area. These sub-functions shape the initial picture of the search environment.

We determine the number of hits on each timestep through the `detection` sub-function. This sub-function receives all of the positional information of both the searchers and the targets from the `search` function as well the design point characteristics. Using this information, it determines the Euclidean distance from the target to each searcher, then utilizes a `sensor` sub-function to determine the probability of detection for each agent based off the distance and the sensor characteristics. Once the `detection` sub-function ascertains

the agents possessing the potential of detecting the target, it calculates which of these succeed. Then the sub-function returns the searchers which successfully detect the target and the searchers which potentially detect the target to the `search` function. This can be expanded further in several ways, though reserved for future work. The primary aspect is the introduction of false alarms where there is a possibility on each timestep for the searcher to generate a detection even if the target is not there which accurately reflects real-world situations. Another extension includes requiring multiple detections for a positive identification of the target representing identification and localization efforts. The `detection` sub-function carries out this detection process and returns to the `search` function which searchers detect the target during each timestep.

The heart of the search pattern resides in the `movement` modules. The searcher and target `movement` modules determine how the searcher and target move during each timestep. These sub-functions receive all information about the model from the `search` function and determine the positional shift of the agents for the next timestep. As seen in Codeblock 2.3, these sub-functions do not update the position of the searchers or targets, but rather simply compute the change in their positions, allowing the search function to perform the update. Furthermore, the sub-functions track the heading of the searchers and targets. By changing how the agents move on each timestep, one can change the search pattern or even the behavior of the agents. Furthermore, since all aspects of the model are passed to the movement sub-functions, agent behavior can change based off of any aspect of the model. For instance, it is possible to have searchers converge on a target if an initial detection is made, or to simulate the target responding to a detection with evasive maneuvering. The `movement` function provides the implementation of the search strategy employed in the simulation model.

```
% Target travel determines the movement vector for the target, and the
% next line adds that vector to the current position of the target,
% updating the target's current location.
[ag_data{2,2}]=target_movement(ag_data{2,2},dp_char);
ag_data{1,2}=ag_data{1,2}+ag_data{2,2}(1,1:2);
```

Codeblock 2.3: The `movement.m` sub-functions generate matrices that contain the change in the x and y direction for every agent over the current timestep and returns this to the `search.m` function. `Search.m` then processes the movement by updating the x and y location of the searchers.

While the functions and sub-functions listed above provide the core of the framework, additional features may be added without disrupting the simulation. For example, we implement a `visual` sub-function into the `search` function as seen in Codeblock 2.4 which provides a graphical, animated representation of the search. While this feature does not directly affect the search outcome, it provides a useful tool in debugging the program as it shows the current target and searcher locations, as well as their intended movement over the timestep. This can be expanded to generate movie clips of simulated target movement for analysis at a later time.

```
    if dispyn==1
        [timestep_star]=visual(dp_char,ag_data,hit);
    end
```

Codeblock 2.4: The `search.m` function calls the `visual.m` function when a graphical representation of the search is desired.

CHAPTER 3:
Experimental Design

Having previously constructed a near real-world situation of interest to help motivate our investigation, this chapter continues by further developing the specific scenario and outlining the factors that compose the parameters of the proposed swarm search models. By implementing several search patterns, we demonstrate the flexibility of the framework in the exploration of different search tactics and swarm characteristics.

3.1 Experimental Design

3.1.1 Real-World Modeling

We begin our research by investigating an abstraction of one potential scenario of interest to USPACOM demonstrating the validity of our framework. The canonical operational scenario involves a single target, e.g., the enemy submarine, moving randomly in a predetermined representative rectangular search area. In our scenario we attempt to estimate the time it takes our searchers to detect the target. This scenario is a simplified representation of searching for a submarine utilizing a swarm of UAVs and relates to a wide variety of situations where a hidden target must be found. Other likely situations where a related scenario would be encountered include receiving intelligence of and attempting to localize a signal of interest in urban operations or a search attempt for a person and/or vessel lost at sea. In the nature of our scenario, we attempt to model the search and localization of a submarine in which we have intelligence on the approximate location of the target.

Alteration of the model may be helpful in gathering insight on several other real-world scenarios of interest to operational commands such as USPACOM. By defining a search region of interest in the operational area and setting our Measure of Performance (MOP) as the number of targets able to enter the area of interest, we are able to model the escort of a high value unit, such as a carrier strike group, through contested waters. This alternate scenario would pose an intrinsically different problem than the previous model as we are attempting to clear an area and then prevent further targets from entering, whereas, in the previous model we are solely concerned with finding the target as quickly as possible.

Other variants of the search problem which may be easily addressed by the proposed framework include modeling search effort required to clear a minefield; for example, by increasing the number of targets in the search area and making them stationary. In this particular model, the primary concern might be the percentage of targets found over a period of time or the time it takes to clear a certain percentage of the targets.

Analysis of the above search scenarios provide the decision maker with valuable information; our framework can quickly adjust from one scenario to the other by using different modules to initialize the target and searcher positions. We focus on the first scenario, localizing a single submarine, for the purpose of our research, and leave the two other proposed models for future work.

3.1.2 Model Construction

In order to construct our proposed swarm search model, we need to develop an abstraction of the real-world scenario. While the specific shape of the search area changes, most search areas are generalized to a rectangle. Therefore, we establish the search area in our model as a rectangular box where the exact dimensions considered are controllable factors with little loss of fidelity. While the search area defines boundaries for the search agents, the target does not possess knowledge of the boundaries and thus travels freely out of and back into the search area. This is typical of searches performed in open waters, although less representative of searches conducted in areas where physical boundaries determine the search area. Note that this latter limitation can be addressed by imposing similar constraints on the target preventing its ability to traverse specific boundaries. In essence, the search area represents the total area that we believe the target to be in and delineates where we concentrate our search efforts.

Since the search area is based on the suspected location of the target, we assume the intelligence is reliable and the target is located within the search area when we begin our search. We further assume there is some uncertainty as to the exact location of the target; the target does not start in the center but rather is (uniformly) randomly located inside of the search area. The target is unaware when it is within detectable distances of the searchers and thus does not change its behavior throughout the scenario. However, the target assumes searchers are attempting to locate and track it, and thus it continues to move in a defensive

manner by randomly altering course to potentially confound attempts to predict its movements. In other words, the target behaves in a semi-intelligent but nonreactive manner.

The starting location of the search agents can be arbitrarily determined as we are able to orient the search box relative to the initial approach to the target area. Therefore, we orient the search area so that the agents begin in the bottom left corner, heading towards the top of the box. Restricting the guide agent to inside of the search area for the Agents Abreast pattern while all agents are restricted to inside of the search area for the Agent in a Column and Random Walk pattern. We also know the endurance of our search agents, providing us with a termination point at which the agents must return to their recovery platform.

3.2 Simulation Construction

As discussed in Chapter 2.6, MATLAB is the primary tool used for modeling this scenario. With the framework in place to utilize the Monte Carlo method to explore different design points, we develop the modules to construct our search area as well as control the agents as they perform their searches. Since the search agents in this study are not explicitly representing specific UXSs, recall that we consider a nominal fixed search speed for all search agents. The first two modules for initial development and implementation include one managing the initial setup of the search area and another dictating the movement of the search agents and targets.

We develop two separate functions to initialize our agents' starting positions, one for the target and another for the searchers. In this work, we initially assume no prior intelligence to prioritize the search effort, but make the framework general enough to allow non-uniform prior probability. Since we assume the target starts at a random location, its starting position is randomly chosen with no bias to any portion of the search area, i.e., with a uniform distribution, as can be seen in Figure 3.1.

Once the location of the target is set, we randomize the target's heading. We initialize the searchers starting locations by setting the guide searcher's location to the bottom left corner of the search area with the remaining agents positioned relative to the guide. For the Agents in Column pattern, the following agents are spaced to the rear of the guide, while for the Agents Abreast and the Random Walk patterns, the agents start abreast of

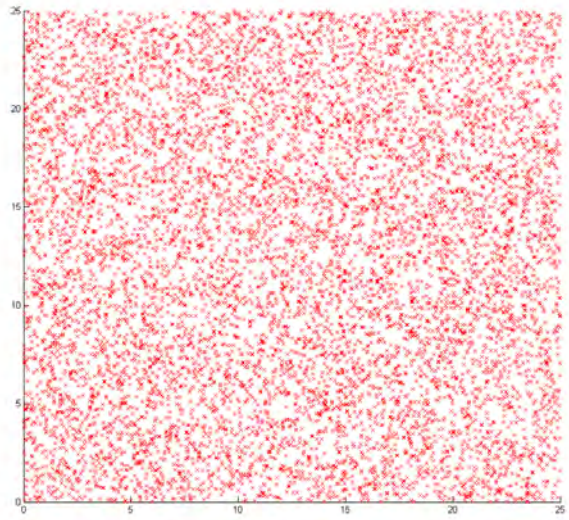


Figure 3.1: Unbiased nature of the target's starting location in a 25 by 25 search area over 10,000 replications. Each red $\times$ represents a starting location with the darker marks representing overlap in the marker location.

each other, extending to the right of the guide's location. In the Random Walk pattern, the initial headings of the searchers are configured towards the center of the search area.

We need to develop another major component for implementation into the MATLAB framework, that is, the `movement` modules. The `movement` modules control where the agents begin the next timestep, and as such, determine the search pattern that the searchers perform as well as the detection avoidance strategy of the target if one exists. As with the initialization functions, we use separate modules for target and searcher movement. This modularity supports the different behaviors being exhibited between the searchers' search patterns and the target's (potentially evasive) maneuvers. As such, these modules are the most logic-intensive; they require tracking of some or all previous state information in most cases. In order to simplify execution of movements, updates to locations are performed using a series of logical checks. The primary check consists of determining if the searchers are inside the search area, which, if successful, then the searchers continue with their associated movement; for both cases of Agents Abreast and Agents in Column, the agents continue to move in the current direction of travel as dictated by the boundary checking logic, whereas for Random Walk, they choose another random direction limited

by heading. If a given searcher is outside of the search area, then we compute its next course of action depending on which boundary it encounters. For example, if the searcher is at the top or bottom of the search area, this involves moving horizontally to continue its search pattern; if it is at the left or right boundaries, movement involves changing direction ensuring the search agent begins its next pass still within the search area. In the case of Random Walk, if the searcher is outside of the search area, it turns a random amount towards the center and moves in that direction. The `target movement` module is identical to the `searcher movement` module for Random Walk with one exception, as it does not change direction towards the center of the search area if the target finds itself outside of the box. Since the `movement` modules are nominally the most complex modules, the fundamental elements of the search patterns employed by the swarm are characterized in these `movement` modules.

3.3 Design of Experiments

We seek to explore how the different factors, such as number of search agents, search speed, detection performance, and their interactions, affect the search capabilities for a given search method. One exhaustive approach to identify the dependence of the search performance on such factors could require varying each factor one at a time; this would allow us to analyze how each factor performs with all other factors at every other possible value. While this may appear to be the most complete method, such an approach quickly expands beyond feasibility for more than a few factors of interest. For example, with eight factors at ten levels each, we would have 10^8 design points not including any replications. By intelligently designing the simulation experiments, we can achieve nearly the same level of statistical fidelity with a fraction of the design points [32], [33]. One method of investigating these factors in the design of experiments (DoE) is utilizing a two-level, or 2^k screening factorial design, which investigates each of k factors at its low and high levels [29]. We can supplement this with a central composite design, which includes the star, or axial, points of the design, and the origin which allows the detection of nonlinearity with minimal additional points, as seen in Figure 3.2. These models allow exploration of all interactions as well as all of the main parameters independently. Unfortunately, utilizing factorial designs may still require significant computational resources as the number of design points still increases exponentially with each additional factor [34]. For instance, when

investigating eight factors for a screening experiment we need $3^8 = 6561$ design points, if we run every design point with 10,000 replications we would need 65,610,000 total executions of the simulation. As our current model takes about one second per replication, that would lead to nearly 759 days to complete the total set of experiments.

Alternatively, we can implement a fractional factorial design as seen in Table 3.1, which assumes that higher order terms are negligible [29]. Based on the resolution of the fractional factorial design, we risk confounding the main effects with those of higher-order terms. For our model a resolution V fractional design would be adequate, causing the main effects to include noise only from the fourth-order terms or higher, as well as the second order terms from the third-order terms or higher. In doing so, we assume the significant interactions to be of the second order or lower, allowing us to significantly reduce the number of design points we would need to generate. Further explanation of fractional factorial designs can be found in Law [29] or Sanchez [35]. We ultimately choose to implement a more efficient, space-filling design, as seen in the following paragraph.

n	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
1	80	80	20	1	0.2	0.05	45	45
2	100	80	20	1	2	0.05	45	90
3	80	100	20	1	2	0.05	45	90
4	100	100	20	1	0.2	0.05	45	45
5	80	80	40	1	2	0.05	45	45
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
63	80	100	40	3	2	0.3	90	90
64	100	100	40	3	0.2	0.3	90	45
65	90	90	30	2	1.1	0.175	67.5	67.5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
79	90	90	30	2	1.1	0.175	45	67.5
80	90	90	30	2	1.1	0.175	67.5	67.5
81	90	90	30	2	1.1	0.175	67.5	45

Table 3.1: Resolution V fractional factorial design with star-points for 8 factors. This design incorporates the high and low values of the factors we utilize in our model. A resolution V fractional factorial design with star-points for eight factors contains 81 design points. We explain these factors in greater detail in Section 3.3.1.[1]

[1]The complete design matrix in comma-separated value format (filename `Resolution_V.csv`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

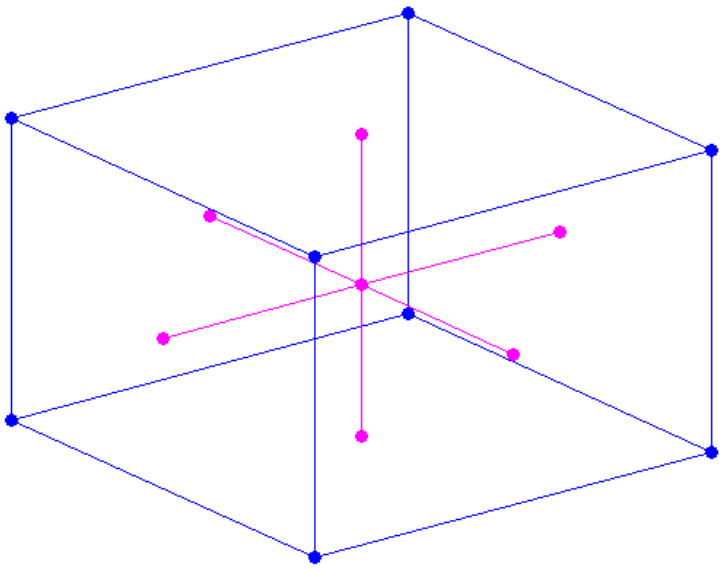


Figure 3.2: Full Factorial Central Composite Design for Three Factors. Full factorial points are in blue with the star, or axial points in magenta, after [35]

This type of screening experimental design is limiting as it only investigates the two extreme levels for each factor; while this is enough to investigate linear models it would not be able to clearly reveal nonlinear effects. In order to capture these potentially nonlinear effects, one can employ a space-filling design. For example, a NOLH design provides us with a space-filling design with relatively few design points [36]. The NPS Simulation Experiments and Efficient Designs (SEED) Center [37] provides a spreadsheet tool for developing NOLH designs for up to 29 factors; this satisfies our need for a space-filling design for the eight factors in our simulation model, as we explain in Section 3.3.1. A resulting design using the NOLH template is partially annotated in Table 3.2.

Having developed our experimental design, we seek to identify the number of replications necessary for each design point. We begin by performing 10,000 replications for multiple design points from each of the three simulation models. We plot the variance against the number of replications, as seen in in Figure 3.3, and analyze how the variance changes as the number of replications increase. In all of the selected design points, the variance

[2]The complete design matrix in comma-separated value format (filename `NOLH_Initial.csv`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

n	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
1	100	81.875	29	1.375	1.81	0.20625	75.9375	66.09375
2	98.125	100	23	1.75	1.2	0.096875	78.75	59.0625
3	97.5	88.75	38	1.3125	0.55	0.198438	77.34375	46.40625
4	91.25	97.5	40	1.8125	1.91	0.089063	81.5625	47.8125
5	98.75	80.625	29	1.4375	1.53	0.229688	63.28125	70.3125
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
29	84.375	85.625	27	1.125	0.88	0.057813	68.90625	71.71875
30	88.125	96.875	36	1.875	1.11	0.26875	46.40625	54.84375
31	83.125	88.125	33	1.25	1.77	0.159375	47.8125	57.65625
32	87.5	96.25	23	1.1875	1.44	0.3	60.46875	56.25
33	83.75	87.5	28	1	0.78	0.073438	64.6875	60.46875

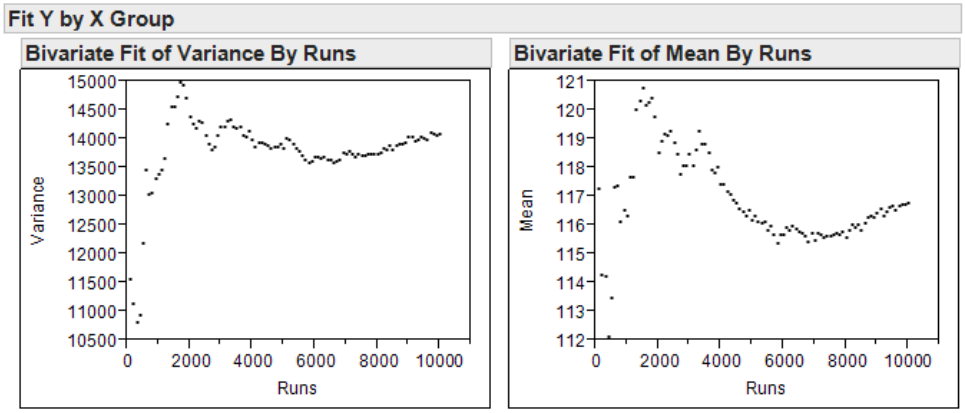
Table 3.2: Nearly Orthogonal Latin Hypercube design for eight factors of interest in the swarm search models investigated in this thesis. We explain these factors in further detail in Section 3.3.1.[2]

fluctuates considerably in the first 2,000 replications but stabilizes by replication 6,000. We identify this as an adequate number of replications to generate a consistent confidence interval around the estimates of our mean [29].

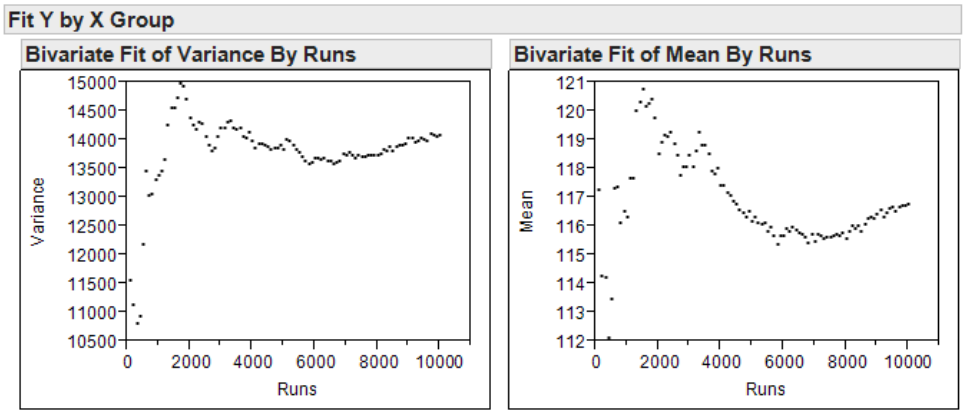
3.3.1 Variables of Interest

Based on our assessment of the simulation, we identify eight key variables which seem likely to influence the responses, namely the search performance. These variables, with their valid ranges and units are defined in Table 3.3 and described further in this section. We also designate the duration of one timestep, which contributes to how several of our factors are implemented despite not changing from design point to design point.

`Timestep Duration` is the length of time that passes during each timestep. We set this to one minute for every scenario, and while remaining constant, it contributes to how several of the other factors are implemented. Foremost, detections occur once at the beginning of every timestep; therefore, our sensors make one detection per minute. If the sensor we implement performs multiple glimpses per minute, then we need to either incorporate that into the `detection` module, or modify other factors. Similarly, as velocity is a function of distance and time, we can shorten the duration of a timestep by either increasing the dimensions of our search area, or by reducing the searchers' and target's velocities.



(A)



(B)

Figure 3.3: This figure shows how the variance and mean change as the number of replications of two design points increases from 100 to 10,000 by intervals of 100. Variance stabilizes at 6,000 replications.

`Height`, `Length` are the dimensions of the search area. We varied the dimensions between 80 and 100 NM providing a search area between 6400 and 10,000 square nautical miles. This size is a reasonable search area since it would encompass an operating area that would take a target traveling eight knots at least ten hours to cross. As reference, we can fully partition the East China Sea into 58 parts, each smaller than the minimum size of our search area.

`NumAgent` is the number of agents that are active in the search. This parameter greatly influences the effectiveness of the search, for example, with increasing numbers providing a wider sweep width when running abreast, or increasing the probability of detection

Variable	Name	Range	Units
X_1	`Height`	[80,100]	nautical miles
X_2	`Length`	[80,100]	nautical miles
X_3	`NumAgent`	[20,40]	agents
X_4	`VelAgent`	[1,3]	$\times$60 knots
X_5	`SpaAgent`	[.5,2]	nautical miles
X_6	`VelTarget`	[.05,.3]	$\times$60 knots
X_7	`TRTarget`	[45,90]	degrees
X_8	`TRAgent`	[45,90]	degrees

Table 3.3: Simulation variables considered in the presented statistical design of experiments. Each of these factors are systematically varied over the defined ranges during several simulation replications.

by having multiple detection opportunities per sweep, or some combination of these two advantages.

`VelAgent` is the search agents speed of advance, notionally assumed to be aerial assets. The low value of one represents transit at 60 kts, while the upper value of three is assumed to represent 180 kts. These are reasonable speeds compared to modern UAVs, and allow an adequate search of the environment. While a higher speed indicates that the area will be searched faster, it can also lead to some areas being searched less thoroughly.

`SpaAgent` determines how spread out, i.e., the inter-agent spacing, the search agents conduct the search. In the case of Agents Abreast and Agents in Column, this spacing remains constant, either laterally or behind the leader. Agent spacing greatly influences the effective sweep width in some scenarios, determining how quickly the search area is covered in the initial pass. Alternatively, if the spacing is too great, then large areas will not be covered as there are gaps in the sweeps between adjacent sweeps.

`VelTarget` determines how fast the target is assumed to travel. Recall the assumption that the target is unaware of the search efforts against it, such that we assume that the target travels at a constant speed. Since the proposed study is highlighting the employment of high speed agents (e.g., UAVs), against a much slower target (e.g., a submarine), we can limit the target's speed to be between 3 to 18 kts. This nominal range relates to speeds commonly found by water bound targets, though this speed is significantly faster than what is commonly found by drifting targets.

`TRTarget`, `TRAgent` reflects the maximum turn rate of the target and searchers. The agents' heading changes are constrained to simulate restrictions to an actual agents ability to change direction. These variables represent the maximum change in degrees from their current course. For example, a turn rate of zero would limit the agent to moving in a straight line, while 180 allows completely random motion. With a lower level of 45 degrees we force the target to maneuver while the upper level of 90 prevents the target from dwelling overly long at any specific point.

3.3.2 Simulation Response Variables

In the previous sections we outline the input parameters and the experimental design for the swarm search models of interest. In this section we discuss the simulation outputs, which represent the measures of performance or response variables that we use for in-depth analysis in the next chapter.

In running these scenarios we are able to obtain four main quantitative values, denoted $Y_1, \ldots, Y_4$, which we used to measure the efficiency and efficacy of given swarm search models as seen in Table 3.4. We expand on these variables throughout this section. The first measure collected by the simulation framework includes the time to first detection, which represents the earliest point in the run when a positive detection is made on the target. We do not include false positives as a detection for our response variable. Note that in order to capture and simulate the finite endurance of the search assets and search mission duration, a simulation time limit is imposed, which leaves the possibility that the searchers make no detections prior to the conclusion of the search mission. As such, in our analysis, the conditional probability of detection for each of the design points is also computed and recorded as we will discuss in Section 4.1.

As an additional feature, the proposed simulation framework also records the number of agents that successfully detect the target throughout the search evolution. While in our given scenarios, such data may not provide significant benefit as the individual simulation run terminates after the first detection occurs; this additional information can be obtained should the termination criterion be based on mission duration, regardless of the number of detections, or in the case that multiple detections are required to perform a positive identification of the target. The final statistic that our framework records is the number

of detection possibilities that occurred. This ground truth information provides several benefits in determining the quality of our sensors. For example, if we see a large number of attempted detections in relation to the number of positive detections, we can determine that the sensor performs poorly for that scenario. Each MOP highlighted above provides insight into different aspects of the search processes. For illustrative purposes highlighted in this thesis, we focus mainly on the time to first detection and probability of detection as our primary metrics.

Variable	Name
Y_1	`Timestep`
Y_2	`Passes`
Y_3	`Hits`
Y_4	`Found`

Table 3.4: Response variables collected in the presented framework. Each of these responses contribute to at least one MOP.

`Timestep` is a positive integer variable that represents the timestep at which the first detection of the target occurred. The average of this variable for a specific design point (i.e., over replications) provides the expected time to first detection for the given input factors. The lower the value of this response variable, the more effective our search; this indicates a more desirable swarm search configuration.

`Passes` is a positive integer variable that represents the number of possible detection of the target. `Passes` increments once per timestep for each UAV that has a non-zero probability of detection during that timestep. This measure provides information on the effectiveness of the sensor, as well as the search process itself; a high number of passes could be the result of a poor sensor or of a search pattern that covers the search area but leaves large portions covered only by parts of the sensor that have poor probabilities of detection.

`Hits` is a positive integer variable that represents the number of actual positive detections on the target. This provides information on the ability to maintain track on a target. In practice, if only a single detection occurs it may be a false alarm, but if multiple detections occur on the target over a relatively short time period and distance, then it is more likely that we actually found the target. Furthermore, by coordinating hits to times and locations, this response can measure the effectiveness of the search pattern to maintain contact and track the target.

`Found` is a binary variable, that is, zero if the target was not found prior to the time limit of the search and one if the searchers successfully found the target. The average value of this variable for a particular design point provides the probability of detection for the given design point's factor values. Higher probability values correspond to favorable configurations of swarm searchers.

Having discussed the design of our experiment and the development of our model, we continue with the conduction of simulations and analysis in Chapter 4.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Analysis and Results

This chapter describes the methods of analysis used to identify quantitative insights leveraging the proposed assessment framework including the constructed search models of interest, designed experiments, and resulting response data. We perform analysis on the data from the three baseline simulated swarm search models, Agents Abreast, Agents in Column, and Random Walk, followed by attempts to reduce the variability in the generated regression models. These studies help to validate our framework as a tool capable of simulating swarm searches and generating analyzable data, as well as highlighting potentially relevant design decisions.

4.1 Analysis Methodology

We concentrate the analysis on one of the measures provided by the simulation, namely `Timestep`, which represents the first timestep the target is found. As such, `Timestep` is a discrete random variable determined for each design point and represented by the average value over 6000 replications. We seek to identify which factors pose the greatest influences on the (expected) time to first detection. To condition the data that allow for actual detection of the target, we exclude those replications where the search agents did not find the target by the allotted mission time, thereby determining the conditional expected time to first detection for each design point. Using this conditional expectation, we perform a standard least squares linear regression to determine the influential factors, thus analyzing main effects, two-way interactions, and quadratic effects (if applicable). Once such influential factors are identified in these screening exercises, we can further investigate sensitivities to these significant factors by fixing the remaining sources of variability at their center values and repeating the simulation experiments. With our new data, we once again perform a standard least square regression to develop our refined regression model. We then determine the parameters which minimize our expected time to detection, either analytically (if possible) or computationally. Finally, based on these search performance prediction models, additional swarm search factor values are used to test the validity of these models and their predictive value.

4.1.1 Regression Analysis

The purpose of our analysis is to determine how one variable is affected by other variables, which is referred to as regression analysis in statistics [38]. Since we utilize a stochastic simulation model, we cannot determine exact deterministic values of measures of performance; by analyzing the relationships between the factors that we assume or identify to be important, we can predict our measure of performance from the aggregated simulation data [39]. One of the most basic and intuitive methods is to develop a linear regression model that attempts to minimize the distance from each data point to the fit. As seen in Montgomery [39], the simplest linear prediction model is of the form seen in Equation 4.1 and involves only a single factor, where Y is our response variable, and β_0 is the y-intercept, and β_1 is the regression coefficient representing the change in the mean of the response variable due to a change in factor x.

$$Y = \beta_0 + \beta_1 x + \varepsilon \tag{4.1}$$

The random error term, ε, is what separates this regression model from a deterministic model, allowing our data points to fall above or below the regression line represented by Equation 4.1. This simple linear model can be expanded to include other independent factors, their interactions, as well as their polynomial terms. We see the resulting general additive form in Equation 4.2, where x_i, referred to as the predictor or regressor variable, represents main effects, the interactions between factors, and polynomial terms in the regression model. In addition, β_i represents the regression coefficient of the corresponding regressor.

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \varepsilon \tag{4.2}$$

As our goal is to find a prediction model that most accurately predicts our measure of performance for any set of factor values, the objective is to identify the coefficients, β_i, in Equation 4.2 which provide us with an acceptable regression fit. In order to do this, we utilize least squares regression. The principle behind least squares regression revolves around choosing values for β_i that minimize the vertical distance each data point lies away from the regression plane [39]. This is accomplished by choosing values for our estimators

of β_i, commonly represented as $\hat{\beta}_i$, that minimize the square of the vertical distance of each point from the regression line, as seen in Equation 4.3.

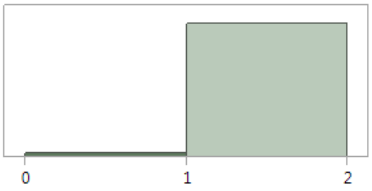
$$f(\beta_0, \beta_1, ..., \beta_k) = \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + ... + \beta_k x_i^k)]^2 \tag{4.3}$$

Once we develop our least squares regression, we can determine the adequacy of statistical model utilizing our coefficient of determination (R^2) and adjusted R^2. While both represent the goodness of fit of our statistical model, R^2 always increases when we add a regressor to our statistical model, regardless of the contribution of the variable, whereas adjusted R^2 only increases if we add a regressor which improves our statistical model. Utilizing adjusted R^2 helps prevent us from overfitting our model by penalizing us for adding terms that are unhelpful [39]. We therefore use adjusted R^2 for our analysis, and refer to it simply as R^2 . This assists us to provide a simple statistical model that provides insight into the scenario of interest.

4.1.2 JMP in Analysis

For our statistical analysis, we primarily utilized JMP versions 10 and 11. This statistical software provides a considerable number of tools, including a wide array for performing the regression analysis. In this section, we discuss how we utilized JMP as a tool to perform our linear regression analysis.

The first step in our analysis in JMP is importing and filtering the data. After importing the comma-separated value data file aggregating the response data from the simulation, we created a histogram to help us isolate the replications where the searchers found the target from those where it went undetected. To do this, we analyzed the distribution of our `Found` response as seen in Figure 4.1 and excluded those replications where the searchers did not find the target. Furthermore, we generated the Cumulative Detection Probability (CDP) using `Timestep` response data as seen in Figure 4.2. Examination of the raw data, such as that seen in Figure 4.3, reveals that additional pre-processing of the response data is necessary before we are able to utilize the aforementioned regression techniques.



Quantiles		
100.0%	maximum	1
99.5%		1
97.5%		1
90.0%		1
75.0%	quartile	1
50.0%	median	1
25.0%	quartile	1
10.0%		1
2.5%		1
0.5%		0
0.0%	minimum	0

Summary Statistics	
Mean	0.9807929
Std Dev	0.1372525
Std Err Mean	0.0003085
Upper 95% Mea	0.9813975
Lower 95% Mean	0.9801884
N	198000

Figure 4.1: Breakdown for the Agents Abreast search pattern for those replications where the searchers found the target, and those where the searchers failed to find the target. For this simulation model the target was found in 98 percent of runs over all design points.

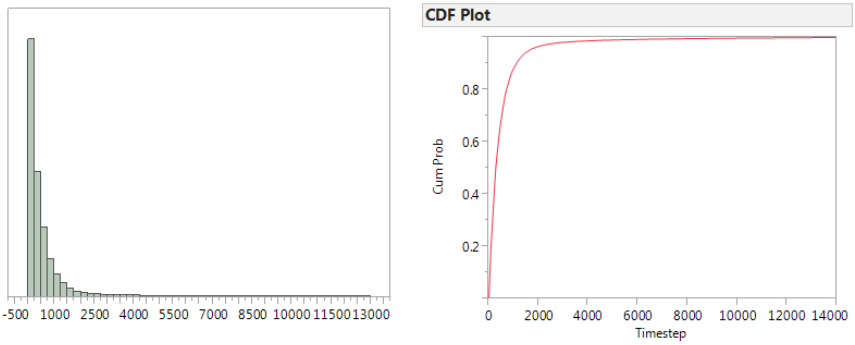


Figure 4.2: This figure shows the breakdown between which timestep the searchers found the target (left), as well as the cumulative detection probability (right).

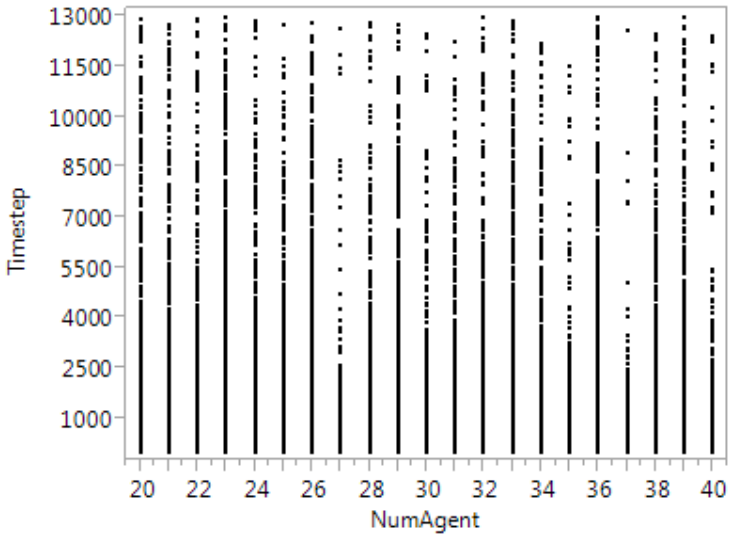


Figure 4.3: Timestep the searchers found the target plotted against the number of agents used for that search. Due to the number of replications and the variability, additional processing is required before these raw data are useable for regression analysis.

Our next step was to compile the raw data (i.e., the responses from the simulation replications) into a useable form. To do this, we consolidated the design points to a single entry each by averaging `Timestep` to find the expected time to first detection for each design point, and creating the MOP, that is, the Probability of Detection (P_D), as provided in the summary table shown in Table 4.1.

DP	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	$\hat{Y}_1$	$\hat{Y}_2$
1	100	81.875	29	1.375	1.81	0.20625	75.938	66.094	536.7	0.982
2	98.125	100	23	1.75	1.2	0.096875	78.75	59.063	601.2	0.992
3	97.5	88.75	38	1.3125	0.55	0.19844	77.344	46.406	436.1	0.99
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.1: JMP's summary table allowed us to summarize our data over each design point. [3]

At this point we begin to leverage JMP's powerful statistical modeling toolbox. From our summary table, we utilized JMP's regression modeling features to perform a stepwise regression to isolate the important factors and interactions that most contribute to the variability in predicting the expected time to first detection. This approach allowed careful iterative selection from all main effects, two-way interactions, and quadratic effects, to a significantly smaller subset as seen in Figure 4.4. JMP allows us to weight our regression coefficients by the amount each design point contributes to the regression model, which is important as we are conditioning on the searchers finding the target. We therefore weight our regression parameters by the P_D, which is a direct correlation to the number of replications where we successfully detected the target within our time limit.

Once we have narrowed our regressors to those significantly impacting the regression model, we utilize JMP to generate a standard least squares regression and begin to screen effects. JMP assists with our effects screening by generating a list of our parameters, sorted by their significance in the regression model as seen in Figure 4.5. We then begin eliminating those regressors with the least significant effects, then iteratively performing the regression until we are satisfied with our regression model (as measured by the adjusted R^2 fit values).

[3] The complete results matrix in JMP data table format (filename `AA_Final_Summary_and_Model.jmp`) can be downloaded from `http://faculty.nps.edu/thchung` under *Software*.

SSE	DFE	RMSE	RSquare	RSquare Adj	Cp	p	AICc	BIC
462610.76	32	120.23554	0.0000	0.0000	.	1	413.1384	415.7314

Current Estimates

Lock	Entered	Parameter	Estimate	nDF	SS	"F Ratio"	"Prob>F"
☑	☑	Intercept	523.669974	1	0	0.000	1
☐	☐	Height	0	1	16850.64	1.172	0.28737
☐	☐	Length	0	1	21488.29	1.510	0.22837
☐	☐	NumAgent	0	1	257503.4	38.919	6.24e-7
☐	☐	VelAgent	0	1	106.1799	0.007	0.93331
☐	☐	SpaAgent	0	1	13624.89	0.941	0.3396
☐	☐	VelTarget	0	1	46632.12	3.475	0.07179
☐	☐	TRTarget	0	1	48767.42	3.653	0.06525
☐	☐	TRAgent	0	1	50.00375	0.003	0.95421
☐	☐	(Height-90.0007)*(Length-89.9965)	0	3	38768.89	0.884	0.46082

(A)

SSE	DFE	RMSE	RSquare	RSquare Adj	Cp	p	AICc	BIC
8037.318	15	23.147812	0.9826	0.9629	.	18	371.4578	341.4299

Current Estimates

Lock	Entered	Parameter	Estimate	nDF	SS	"F Ratio"	"Prob>F"
☑	☑	Intercept	264.914956	1	0	0.000	1
☐	☑	Height	3.81607457	4	26410.05	12.322	0.00012
☐	☑	Length	4.29372432	4	26767.49	12.489	0.00011
☐	☑	NumAgent	-14.532248	6	266349.5	82.848	1.2e-10
☐	☐	VelAgent	0	1	53.89156	0.095	0.76305
☐	☑	SpaAgent	41.2982596	5	20869.78	7.790	0.00086
☐	☑	VelTarget	486.087919	3	45973.44	28.600	1.88e-6
☐	☑	TRTarget	-2.8565885	5	51645.59	19.277	4.74e-6
☐	☐	TRAgent	0	1	18.30853	0.032	0.86067
☐	☑	(Height-90.0007)*(Length-89.9965)	0.65003468	1	2176.898	4.063	0.06212

(B)

Figure 4.4: Stepwise regression process in JMP. Initially, only factor included in model is the y intercept (A). As stepwise regression progresses, factors are added and removed to the regression model to find the best fit. The final regression model (B) with R^2 at the top, and the factors that were considered to impact the regression model checked below as well as their predicted regression coefficients.

Furthermore, we can explore the effect each factor has on our resulting prediction model utilizing JMP's prediction profiler, as seen in Figure 4.6. This tool also allows us to see how the prediction of the response variable changes as we adjust the values of the regressors. The prediction profiler also provides us with 95% prediction intervals for the response variable of interest. This also provides a visual representation of how the response variable will change as we adjust the value of a regressor across its possible values. The magnitude of the slope of the regressors indicates the influence that regressor has on the predicted response variable, provided the remaining variables remain constant. For example, if we set the `Height` and `Length` of the search area to 90 NM, utilize 40 agents in the search,

Term	Estimate	Std Error	t Ratio	Prob>|t|
NumAgent	-14.53225	0.688301	-21.11	<.0001*
TRTarget	-2.856589	0.304756	-9.37	<.0001*
VelTarget	486.08792	54.79596	8.87	<.0001*
Length	4.2937243	0.684591	6.27	<.0001*
Height	3.8160746	0.685101	5.57	<.0001*
SpaAgent	41.29826	9.122192	4.53	0.0004*
(NumAgent-30.0902)*(NumAgent-30.0902)	0.6167171	0.204904	3.01	0.0088*
(Length-89.9965)*(NumAgent-30.0902)	0.8174473	0.278347	2.94	0.0102*
(NumAgent-30.0902)*(VelTarget-0.17419)	-73.18622	27.47667	-2.66	0.0177*
(Height-90.0007)*(Length-89.9965)	0.6500347	0.322498	2.02	0.0621
(NumAgent-30.0902)*(SpaAgent-1.25021)	-6.778918	3.534881	-1.92	0.0744
(Height-90.0007)*(SpaAgent-1.25021)	-4.863032	2.538202	-1.92	0.0746
(SpaAgent-1.25021)*(TRTarget-67.5891)	1.8414936	0.992231	1.86	0.0832
(Length-89.9965)*(SpaAgent-1.25021)	-7.119035	4.262146	-1.67	0.1156
(Height-90.0007)*(TRTarget-67.5891)	0.1235535	0.079631	1.55	0.1416
(NumAgent-30.0902)*(TRTarget-67.5891)	0.0362142	0.063874	0.57	0.5791
(VelTarget-0.17419)*(TRTarget-67.5891)	-3.514209	6.400406	-0.55	0.5910

(A)

Term	Estimate	Std Error	t Ratio	Prob>|t|
NumAgent	-14.73613	0.896082	-16.45	<.0001*
TRTarget	-2.841176	0.396839	-7.16	<.0001*
VelTarget	489.04154	71.4936	6.84	<.0001*
(SpaAgent-1.25021)*(VelTarget-0.17419)	-1025.027	204.1395	-5.02	<.0001*
Length	4.2551615	0.892417	4.77	<.0001*
Height	3.8632677	0.891434	4.33	0.0002*
SpaAgent	42.338716	11.90374	3.56	0.0016*
(Length-89.9965)*(SpaAgent-1.25021)	8.7803036	2.654482	3.31	0.0030*

(B)

Figure 4.5: Effect Screening in JMP. Initial (A) and final (B) regression models developed from stepwise regression in JMP. Parameters are sorted by their contributions to the regression model. The blue line denotes 0.05 significance level, orange values are those parameters with significance less than .01, red denotes parameters with significance between .01 and .05, and black for those above .05.

spaced 3 NM apart, we expect to find the target at 450 timesteps, or 7.5 hours. Furthermore, we can see if we hold all other regressors constant, and reduced the number of agents to twenty, we would expect to find the target in about 750 timesteps, or 12.5 hours.

4.2 Simulation Model Analysis

Having discussed our analysis techniques, we continue with the analysis we perform on our three cases, Agents Abreast, Agents in Column, and Random Walk. In this section, we explain the analysis and the generating process for our statistical models. We conclude with a discussion on how we arrive at our prediction models from our simulation model.

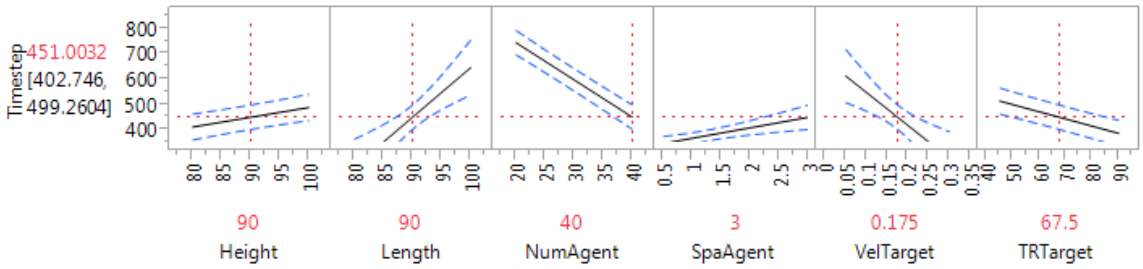


Figure 4.6: JMP's Prediction Profiler allows us to experiment with how different values for our regressors impacts the response variable.

4.2.1 Agents Abreast

Recall that the first step in our analytic process is to aggregate the resulting data in a format conducive to analysis. To do this, our framework generates two output files per simulation. The first file is a .mat file, which is a MATLAB structure file that includes the specific variable values for each design point, the response variables, and the settings of our random number generator, including the seed. This saved MATLAB file serves as a backup to retrieve the simulation results for one or more replications without repeating execution of the simulation models. The second file that the framework generates, as described in previous sections, is a comma-separated value, which contains all relevant response and design point data.

As discussed in Section 4.1, we perform a standard least squares analysis to determine the influential factors in the regression model. When the initial results returned, the expected time to first detection seemed biased to the right. Upon further investigation, we realize that the program set the time of first detection as the termination of the replication if the searchers did not find the target, causing significant probability mass to accumulate on the final timestep. To account for this, we exclude replications where the searchers do not find the target and condition our results on finding the target. For example, for design point two (see Table 4.2), the expected time to first detection is 601 timesteps, given that the target is detected on that run, and the probability of detection is 0.992 as seen in Figure 4.7.

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8
Design Point 2	98.125	100	23	1.75	1.2	0.096875	78.75	59.0625

Table 4.2: Design Point Two

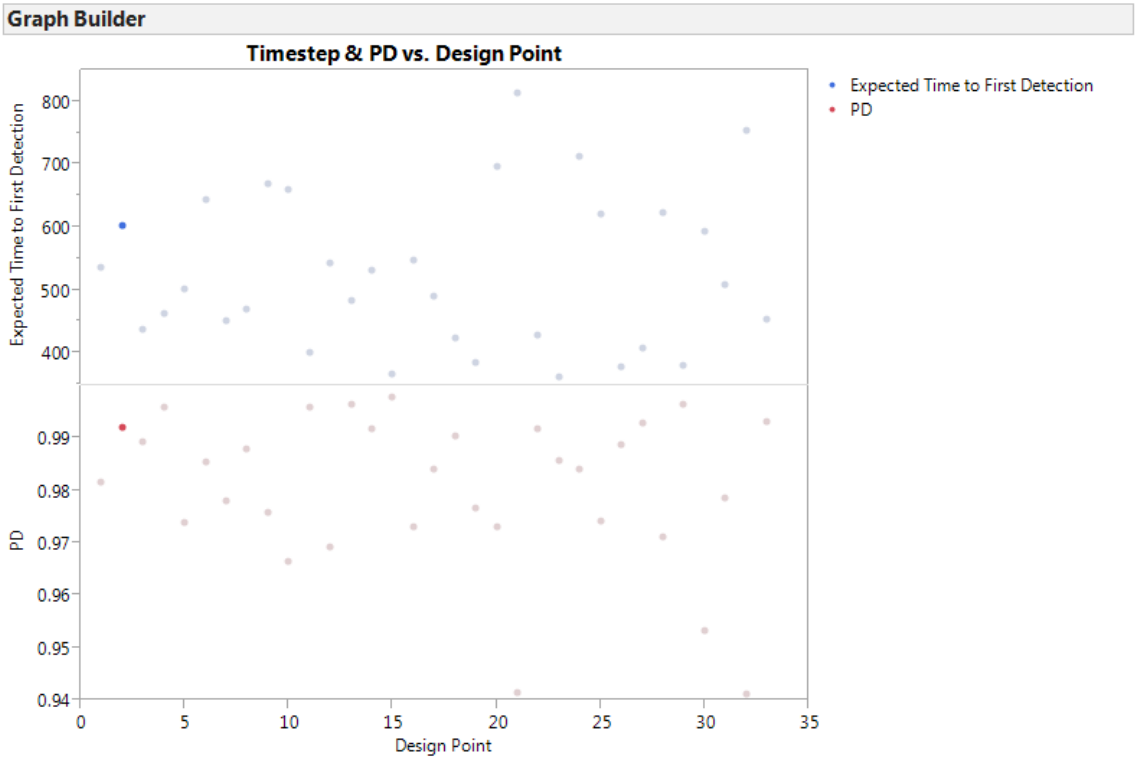


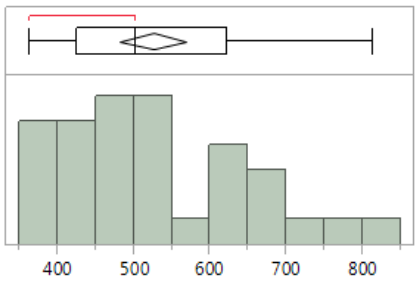
Figure 4.7: Expected time to first detection and probability of detection for Agents Abreast search model by design point. The highlighted data point corresponds with design point two. Blue represents the time to first detection. Red represents the probability of detection.

To develop our statistical model from this processed data, we create a summary table in JMP. This allows us to consolidate the information from each design point, determining the expected time to first detection, variance of the time to first detection, and the probability of detection for each design point. We perform a quick visual inspection of our data, and notice no significant outliers in the expected time to first detection, but two outliers for our P_D as seen in Figure 4.8. These outliers correspond to design points possessing a high target velocity, and a low target turn rate. The outliers do not significantly impact our results as we condition or expected time to first detection on finding the target and all of our design points achieve a greater than 90% detection rate. Furthermore, we weight the impact each design point has on our regression by the probability of detection for the design point. We also inspect the multivariate graph of expected time to first detection and P_D, noticing some distinct correlation between several factors, such as the number of searchers and the expected time of first detection, as seen in Figure 4.9. Performing a visual inspection of our

initial data provides us with an idea of what we should expect from our regression analysis, helps us find possible problems with our simulation model, and potential problems we may encounter when creating our regression model.

Having performed our visual analysis of our results, we utilize JMP's stepwise regression feature to begin the statistical analysis. Weighting the impact the data has on the regression model by P_D, we generate initial regression models of the expected time to first detection. Recognizing that JMP's stepwise regression tends to overfit the statistical model, we can manually determine which additional factors to remove. The initial stepwise regression identifies a statistical model that provides an excellent fit, with an adjusted R^2 of 0.96, but includes six of the main factors, and eleven interaction terms. Furthermore, a significant number of those regressors either have a minor impact on the regression model or there is a reasonable probability that the true value of the regression coefficient is zero as seen previously in Figure 4.5. We remove terms from this initial regression model in order to develop a more manageable product (i.e., fewer terms) while monitoring the goodness of fit. For the Agents Abreast regression model, we only reduce the number of significant factors down to the `Height` and `Length` of the search area, the spacing of the searchers (`SpaAgent`), the velocity of the searchers (`VelAgent`), and the number of searchers (`NumAgent`). When attempting to remove additional factors in our regression model such as `Height` or `Length`, we notice a significant drop in the R^2 from 0.94 to 0.82. This leads us to believe that while these regressors only contribute about eight percent each, they are necessary for the overall accuracy of the regression model.

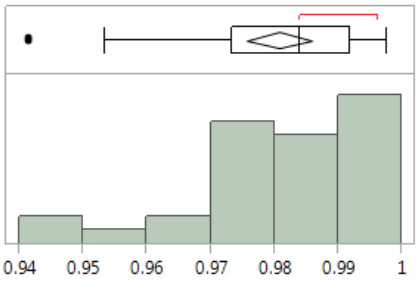
We now try to minimize the effect the noise has on the resulting statistical model. To do this, we modify the experimental design to isolate the factors that we determine to be important. By holding the noise factors constant (at their center values) and utilizing a refined NOLH design for the remaining variables, we generate an additional set of design points to run and collect data. Once the results are compiled, we import them into JMP, perform the linear regression as before, and determine more accurate predictors for the model parameters. This leads us to the estimated or fitted response variable, $\hat{Y}_1$, from the regression model as seen in Equation 4.4. Recall from Section 3.3.1 that X_1, X_2, X_3, X_5, X_6, and X_7 correspond to the `Height` and `Length` of the search area, the number of searchers,



Quantiles		
100.0%	maximum	814.001
99.5%		814.001
97.5%		814.001
90.0%		705.686
75.0%	quartile	621.63
50.0%	median	502.477
25.0%	quartile	425.18
10.0%		377.978
2.5%		362.437
0.5%		362.437
0.0%	minimum	362.437

Summary Statistics	
Mean	524.94357
Std Dev	121.96436
Std Err Mean	21.231271
Upper 95% Mea	568.19026
Lower 95% Mean	481.69689
N	33

(A)



Quantiles		
100.0%	maximum	0.99783
99.5%		0.99783
97.5%		0.99783
90.0%		0.9962
75.0%	quartile	0.99192
50.0%	median	0.98417
25.0%	quartile	0.97342
10.0%		0.95853
2.5%		0.94117
0.5%		0.94117
0.0%	minimum	0.94117

Summary Statistics	
Mean	0.9807929
Std Dev	0.0146094
Std Err Mean	0.0025432
Upper 95% Mea	0.9859732
Lower 95% Mean	0.9756126
N	33

(B)

Figure 4.8: Distribution of Agents Abreast expected time to first detection (A) and P_D (B). There are two outliers for the probability of detection.

the spacing between searchers, the velocity of the target, and the turn rate of the target, respectively.

$$\hat{Y}_1 = 217 + 3.8X_1 + 4.7X_2 - 16X_3 + 43X_5 + 525X_6 - 2.7X_7 + 1.3(X_3 - 30)^2 \qquad (4.4)$$

Notice that the coefficients for X_3 and X_7 are negative, and intuitively this makes sense, for as the number of searchers increases, we effectively have a single combined sensor with a nominally wider detection curve allowing a quicker coverage of the search area which thus decreases the average time to initial detection. Conversely, the coefficient for X_5 is negative indicating that as searchers spread farther apart, the expected time to first detection increases. This effect is simple to visualize for increased spacing as the centers of the detection curves shift further apart as seen in Figure 4.10, but a similar effect occurs as velocity increases because detections occur only at the beginning of each timestep, and not at every point between the searchers' location at the start of one timestep and the next.

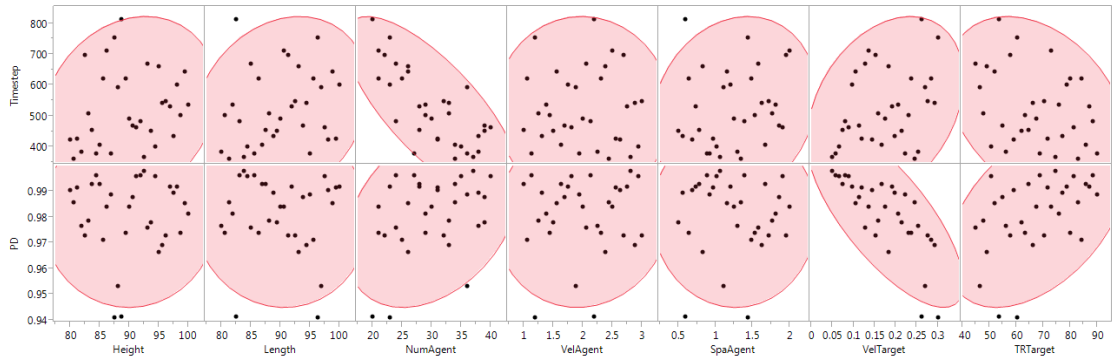


Figure 4.9: Multivariate Scatterplots of Agents Abreast simulation model, showing expected time to first detection (top) and probability of detection (bottom). Density ellipses provide a visual representation of correlation between factors.

This issue is an artifact of the discretization of time and space in the simulation model, and though appropriate for the class of search models investigated herein, may be addressed in future refinements of the proposed framework. Similarly, as agents move closer together, their sweeps overlap, providing increased coverage in that area as seen in Figure 4.11. This turns out to not be an additive increase, as the probability of at least one detection is the complement to the probability of neither searcher detecting the target.

The positive coefficient on the quadratic terms for number of agents indicate that there is a diminishing return to increasing the number of agents beyond a certain point and in fact increasing them too far can have a negative impact on performance. This agrees with intuition to an extent, for as we increase the number of searchers we expect to hit a point of diminishing returns; based on the search pattern, adding additional agents increases the width of the detection curve thus providing additional coverage of the search area but does not reduce the effectiveness of the search anywhere. As such, we should never reach a point where adding additional searchers increases the time to first detection. Due to attempting to fit the statistical model with only main and quadratic effects, and two-way interactions, we cannot fully capture how diminishing returns effects our model. Utilizing $\frac{1}{X_3}$, the reciprocal of the number of agents, could provide a better representation, but as our statistical model is adequate over the range we are investigating, we leave this to future work. Referring back to Figure 4.9, we see that our prediction model is consistent with our initial hypotheses on the dependence of swarm search performance on the number of searchers.

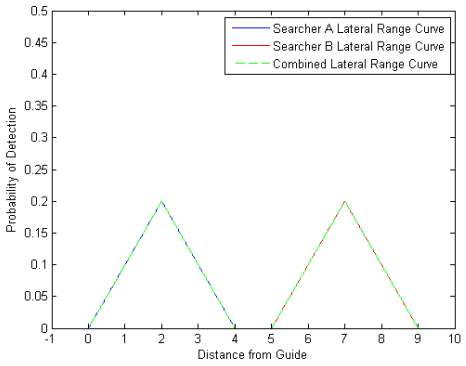


Figure 4.10: Searcher spacing and searcher velocity create gaps in sensor coverage when increased beyond a threshold.

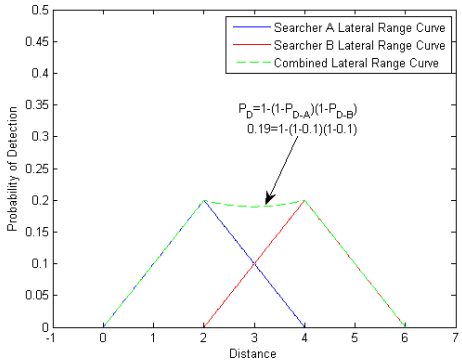


Figure 4.11: Searcher spacing and searcher velocity provide overlap if spaced close enough together, effectively extending into a single detection curve.

Having generated a statistical model, we attempt to validate our work by generating a test design point, performing the necessary replications at the design point, then evaluating how accurately our statistical model predicts our response variable at the design point. We decide to try to optimize the factors to generate the minimal expected time to first detection. To do this, we minimize Equation 4.4, constrained by the valid ranges of values for the factors (refer to Table 3.3). Since the majority of our regressors are main effects only, we are able to simply maximize or minimize them based off their coefficients, such as minimizing the `Height` and `Length` of the search area due to their positive coefficient, and maximizing the turn rate of the target due to its negative coefficient. Since the number of agents consists of both the main and quadratic effect, we perform additional calculations to identify the point where the function is optimized. We take the partial derivative of the function with respect to X_3, the number of agents, and set this to zero as seen in Equation 4.5. Solving for the number of agents we arrive 36.15 agents, which we round to 36 as we are unable to have partial agents. The design point we utilize is summarized in Table 4.3, shown towards the end of this section.

$$\frac{\partial}{\partial X_3}\hat{Y}_1 = \frac{\partial}{\partial X_3}\left(217+3.8X_1+4.7X_2-16X_3+43X_5+525X_6-2.7X_7+1.3(X_3-30)^2\right) \quad (4.5)$$

$$0 = -16+2.6(X_3-30)$$

$$X_3 = 36.15 \quad \sim 36 \text{ search agents}$$

4.2.2 Agents in Column

For our searchers moving in a column scenario we performed our analysis in much the same manner. Upon conducting repeated simulation, we export the resulting response data to a comma-separated value data file to import into JMP. We filter those data to exclude the replications where the searchers did not find the target within the timeframe before performing the initial regression analysis. Upon visual inspection of the scatterplot matrix in Figure 4.12, we expect the velocity and spacing of the searchers to be the main factors. Similar to the previous Agents Abreast regression model analysis, we find JMP appeared to overfit the stepwise regression model with an R^2 of 0.98, and thus we manually reduce the number of parameters in the regression model to produce a manageable size while maintaining an adequate fit, as seen in Figure 4.13, ultimately arriving at a R^2 of 0.92. Having determined the key contributors to the variability, we perform the simulations while holding the noise factors constant at their center values, analyzing the remaining significant factors. As done in Section 4.2.1, this analysis leads us to Equation 4.6 as the statistical prediction model for the expected time to first detection for the Agents in Column simulation model, where we recall that X_4 and X_5 represent searcher velocity and searcher spacing, respectively. The linear and quadratic nature of these factors intuitively makes sense for the same reasons as discussed in Section 4.2.1.

$$\hat{Y}_1 = 5504 - 858X_4 - 1490X_5 + 387(X_4-2.0)^2 + 1149(X_5-1.1)^2 \quad (4.6)$$

Notice that the number of searchers does not have a significant effect on the prediction model for this swarm search configuration of Agents in Column. We suspect this is primarily due to already reaching the diminishing return point of number of searchers at our minimum number of twenty search agents. This is most likely due to the relatively small

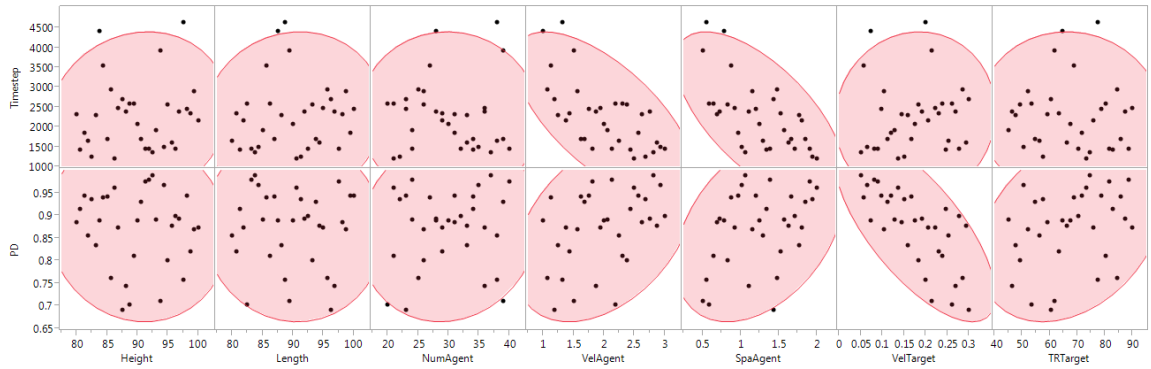


Figure 4.12: Multivariate Scatterplots of Agents in Column simulation model expected time to first detection (top) and probability of detection (bottom). Density ellipses provide a visual representation of correlation between factors.

Term	Estimate	Std Error	t Ratio	Prob>|t|
VelAgent	-914.5841	34.86264	-26.23	<.0001*
SpaAgent	-1211.616	46.63917	-25.98	<.0001*
(VelAgent-2.02227)*(VelAgent-2.02227)	654.10738	83.23894	7.86	<.0001*
(VelAgent-2.02227)*(SpaAgent-1.26665)	655.39752	92.39418	7.09	<.0001*
(VelTarget-0.16997)*(VelTarget-0.16997)	-24877.23	5755.071	-4.32	0.0003*
Height	14.48413	3.459292	4.19	0.0004*
Length	11.705941	3.464975	3.38	0.0028*
VelTarget	925.99592	280.8479	3.30	0.0034*
(SpaAgent-1.26665)*(SpaAgent-1.26665)	361.13069	143.3673	2.52	0.0199*
TRTarget	-3.436317	1.547146	-2.22	0.0375*
(Height-89.9569)*(Length-89.975)	0.3650222	0.551642	0.66	0.5154

(a)

Term	Estimate	Std Error	t Ratio	Prob>|t|
VelAgent	-921.4423	68.67798	-13.42	<.0001*
SpaAgent	-1206.787	92.15255	-13.10	<.0001*
(VelAgent-2.02227)*(SpaAgent-1.26665)	716.82668	159.4986	4.49	0.0001*
(VelAgent-2.02227)*(VelAgent-2.02227)	429.43029	132.4675	3.24	0.0031*

(b)

Figure 4.13: Contribution of regressors in our linear regression model. The stepwise regression overfits the model (a), by removing minimally contributing factors we creating a simpler model (b).

change in target position during each timestep, giving each searcher nearly the same probability of detecting the target. This effect makes intuitive sense as at a 0.1 probability of detection, with twenty agents having possible detections collectively increases this probability to approximately 0.88 probability of detection on each pass, whereby increasing to thirty agents only increases this to 0.96, and increasing the number of searchers to forty agents leads to 0.99 probability of detection. We maintain the range of searchers between twenty and forty agents to provide parity between the parameters of our search models.

We expect that by splitting the search area into several sections, and assigning a subset of available searchers to each subsection, we can greatly improve the search, but leave this to future work.

Once again, we attempt to validate our work by generating a test design point, then evaluating how accurately the computed statistical model predicts the response for the given design point. We determine the design point to minimize the time to first detection in the same manner as we did for the Agents Abreast prediction model. This time, since the regression model possesses two quadratic terms, we need to take the partial derivative with respect to each regressor, once to determine the optimized velocity of the agents, and again to determine the optimized inter-agent spacing. The predicted optimal velocity of the searchers falls outside of the upper range for that variable at 3.1, so we utilize a searcher velocity of 3 kts. Conversely, the optimal spacing of the agents falls well within the range of 0.5 to 2 NM at 1.75 NM. The design point we utilized is seen in Table 4.3.

In Figure 4.14 we show how the negative parameter for the main effect and the positive parameter for the quadratic effect of the X_4 term (that is, the searcher velocity) suffers diminishing returns as searcher velocity increases. Furthermore, the rate of decrease remains constant, regardless of the searcher spacing utilized for the search, as there are no interactions in our regression model. The effect of increasing the searcher spacing also diminishes as they spread further apart.

4.2.3 Random Walk

We generate the Random Walk statistical model utilizing the same process as performed for both Agents Abreast (Section 4.2.1) and Agents in Column (Section 4.2.2).

From initial visual inspection of the scatterplot seen in Figure 4.15, we expect the number of searchers, the velocity of the searchers, and the velocity of the target to be the primary contributors to the variability in our regression model. The regression analysis highlights the same problem of overfitting experienced previously, but we reduce the complexity to an acceptable level and isolate the noise elements as done before. Similar to the Agents Abreast regression model, we find that most of the factors are significant, and are only able to isolate one factor, the spacing of the searchers, as insignificant to the regression model as seen in Figure 4.16. Following the isolation of the noise, we perform the simulation with a

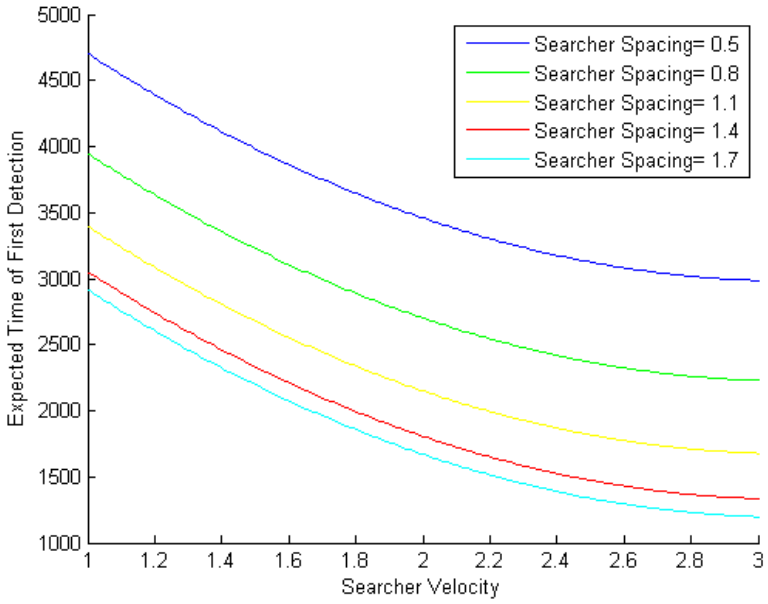


Figure 4.14: Predicted impact that searcher spacing has on the expected time to first detection for the Agents in Column regression model.

refined experimental design specifically to perform sensitivity analysis and adjust the prediction model accordingly. We arrive at Equation 4.7 as the computed statistical prediction model for the Random Walk simulation model, where X_2, X_3, X_4, X_6, and X_8 represent the length of the search area, the number of searchers, the velocity of the searchers, the velocity of the target, and the turn rate of the searcher respectively.

$$\hat{Y}_1 = 529 + 5.3X_2 - 17X_3 - 118X_4 + 879X_6 + 2.4X_8 \tag{4.7}$$

In Figure 4.17 we show how the negative parameter for the main effect of the X_4 term (that is, the searcher velocity) remains constant as searcher velocity increases. Furthermore, the rate of decrease remains constant, regardless of the number of agents utilized for the search, as there are no interactions in our regression model. The effect of increasing the number of searchers also remains constant as more searchers are utilized. We believe this indicates that we have not reached the point of diminishing returns.

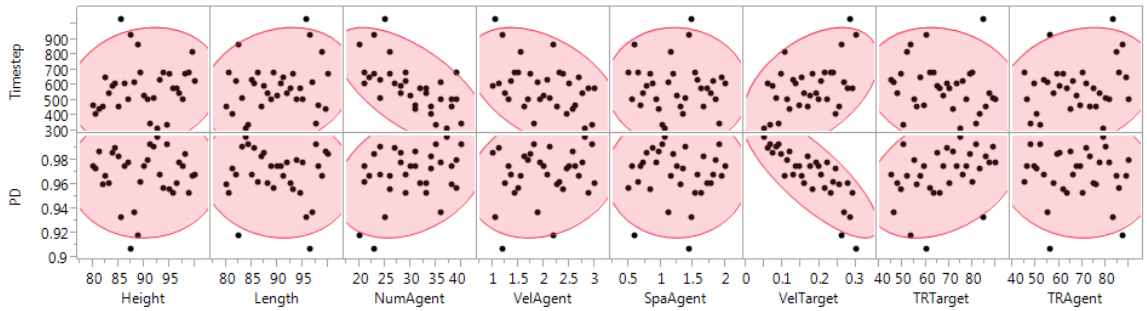


Figure 4.15: Multivariate Scatterplots of Random Walk model expected time to first detection (top) and probability of detection (bottom). Density ellipses provide a visual representation of correlation between factors.

4.2.4 Measures of Performance

Having developed a statistical model for each of the three swarm search patterns and having determined associated prediction models, we are able to extract the measures of performance of interest from the validation results for the optimized factor values for each search pattern. Utilizing JMP to analyze the response data sets, we calculate the expected time to first detection for the predicted optimal design point of each search pattern. For Agents Abreast and Agents in Column, we are within 95 percent prediction intervals, and the Random Walk results fall within twenty timesteps of the upper limit of the prediction interval. While still outside of a 95 percent prediction interval, these results are within the 99.5 percent prediction interval. The factors, the predictions, and the resulting response data of the three prediction design points are seen in Table 4.3.

Search Pattern	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	$\hat{Y}_1$	PI	Y_1
Agents Abreast	80	80	36	2	0.5	0.05	90.0	67.5	185	$\pm$ 82	257
Agents in Column	90	90	40	3	1.75	0.175	67.5	67.5	1102	$\pm$ 392	1108
Random Walk	90	80	40	3	1.25	0.05	67.5	45	87	$\pm$ 145	247

Table 4.3: Validation study using statistical prediction models with optimized design points for the three swarm search patterns.

Term	Estimate	Std Error	t Ratio	Prob>|t|
NumAgent	-16.53734	0.659799	-25.06	<.0001*
VelTarget	948.57919	52.68436	18.00	<.0001*
VelAgent	-116.1905	6.570565	-17.68	<.0001*
TRAgent	2.6011253	0.292107	8.90	<.0001*
Length	5.2424905	0.658636	7.96	<.0001*
TRTarget	-2.00055	0.292646	-6.84	<.0001*
Height	3.5986757	0.656668	5.48	<.0001*
(VelAgent-2.00263)*(TRAgent-67.4766)	-2.519366	0.579655	-4.35	0.0006*
(Length-89.9903)*(VelAgent-2.00263)	-5.593326	1.71113	-3.27	0.0052*
(VelAgent-2.00263)*(VelAgent-2.00263)	62.794987	20.59688	3.05	0.0081*
SpaAgent	-17.30153	8.763156	-1.97	0.0670
(NumAgent-30.1005)*(NumAgent-30.1005)	0.4071074	0.232084	1.75	0.0998
(Height-90.0014)*(NumAgent-30.1005)	0.426428	0.275332	1.55	0.1423
(SpaAgent-1.25043)*(TRAgent-67.4766)	-0.823875	0.586766	-1.40	0.1807
(Length-89.9903)*(Length-89.9903)	0.2108205	0.161735	1.30	0.2121
(NumAgent-30.1005)*(VelTarget-0.17371)	5.6482637	10.36881	0.54	0.5939
(Length-89.9903)*(TRTarget-67.6029)	-0.010911	0.057145	-0.19	0.8511

(a)

Term	Estimate	Std Error	t Ratio	Prob>|t|
NumAgent	-16.4891	0.805374	-20.47	<.0001*
VelTarget	941.48968	64.28988	14.64	<.0001*
VelAgent	-115.6888	8.021892	-14.42	<.0001*
TRAgent	2.6050533	0.35672	7.30	<.0001*
Length	5.313119	0.802561	6.62	<.0001*
TRTarget	-1.959035	0.35678	-5.49	<.0001*
(VelAgent-2.00263)*(VelAgent-2.00263)	81.398078	16.58757	4.91	<.0001*
(NumAgent-30.1005)*(NumAgent-30.1005)	0.815984	0.166791	4.89	<.0001*
Height	3.6903862	0.801327	4.61	0.0001*
(VelAgent-2.00263)*(TRAgent-67.4766)	-3.106259	0.676124	-4.59	0.0001*

(b)

Figure 4.16: Contribution of regressors in the resulting linear regression model. The stepwise regression overfits the regression model (a), by removing minimally contributing factors we create a simpler regression model (b).

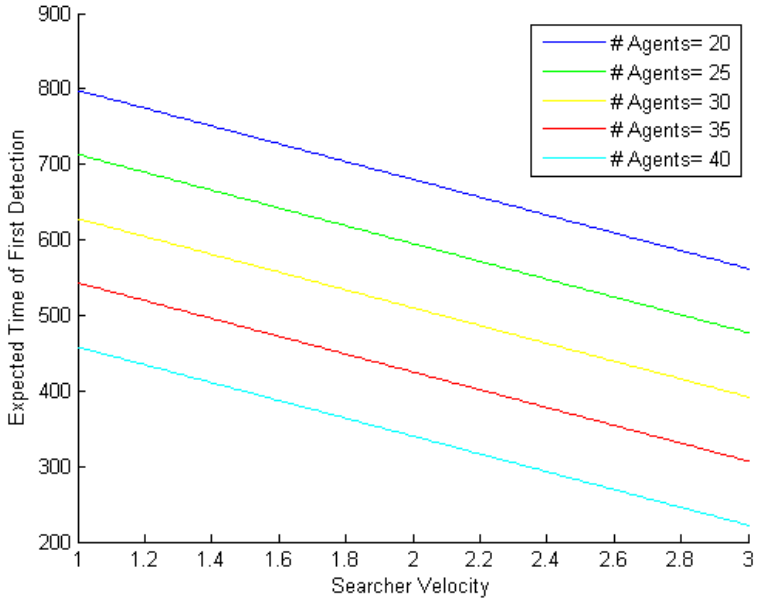


Figure 4.17: Predicted impact that the number of agents has on the expected time to first detection for the Random Walk regression model.

CHAPTER 5:
Conclusions and Recommendations

This section provides an assessment of the presented framework based on the model formulations and statistical analysis completed in the previous chapters. We discuss areas that the models could be improved or changed. Finally, the chapter concludes with proposals for operational applications, follow-on studies, and future applications of utilizing this simulation framework for assessing future swarm search models.

5.1 Conclusions

We created a framework to model, implement, and assess swarm search models in MATLAB in order to perform rigorous statistical analysis and generate potential operational and technical insights. Such a framework is likely to be useful in exploring a wide variety of relevant aspects of swarm search, such as different search strategies and patterns as well as different target distributions and behaviors. Furthermore, the methodology we develop and demonstrate in this thesis can easily be repeated for, e.g., different detector model or search agent characteristics, to specifically analyze how the different sensor or searcher affects the model. As such, the simulation and assessment framework we develop in this thesis provides a useful tool in analyzing search strategies, searcher characteristics, and sensor characteristics that may help inform future technology and operational design and/or employment of swarm search capabilities.

One of the issues encountered during initial attempts to generate a statistical model was overfitting. We attempted to analyze up to three-way interactions, and while the statistical model accurately fit the data, it included every main effect and nearly every two- and three-way interaction. As such, the resulting statistical model indicated that everything was statistically important and did not provide useful analysis of what factors contribute to the expected time to initial detection. Therefore, we limit the studied statistical models to two-way interactions and quadratic effects, allowing us to provide a more useful analysis of the decision variables. We compare the adequacy of both statistical models, and suffer a minor reduction to R^2 of less than 0.001. We conclude that the higher order interactions do not contribute significantly to the statistical model and thus remove them.

5.1.1 Deterministic Search Patterns

The Agents Abreast statistical model is shown to be effective in predicting the expected time to first detection for the Agents Abreast simulation model. For the Agents Abreast statistical model, the number of searchers, is identified to be the dominant factor, which is consistent with our expectation. For the Agents Abreast pattern, we effectively utilize a single searcher with a significantly expanded detection curve. The more agents conducting the search, the greater the effect on the detection curve, and thus the greater the rate of coverage during the search, which corresponds to the negative coefficient for the number of searchers in the prediction equation (refer to Equation 4.4). The larger effective sweep width allows the searchers to cover the search area more quickly than the Agents in Column and Random Walk search patterns, but with a lower chance of detecting the target. Nonetheless, the Agents Abreast pattern completes and repeats its coverage pattern of the entire search area enough times to reach the point of diminishing returns gained from multiple passes faster than Agents in Column model.

The Agents in Column statistical model also is shown to perform well at predicting the expected time to first detection of its simulation model. For the statistical model, the searcher spacing, and velocity main and quadratic effects are seen to be important. While not immediately apparent, we find that these effects match our expectations. As searcher spacing and velocity increase, the formation covers the search area faster, reducing the expected time to first detection which corresponds to the negative coefficient for the main effects (refer to Equation 4.6). Conversely, the quadratic terms are positive, indicating that the benefit gained from increasing speed and spacing diminishes, and eventually will start to hurt the search pattern's performance. While not immediately apparent, we believe this is from the gaps in coverage caused by a poorly organized deterministic search which we discuss in the following paragraph. While surprised that the number of searchers does not impact the performance of the search model, we determine this is due to having surpassed the point of diminishing returns by twenty agents. For similar reasons as those discussed in Section 4.2.1, utilizing more agents for the search can only improve our results, but the benefit gained by each additional agent declines significantly as we already get over twenty looks at the target during each pass. We believe our statistical model to be an accurate representation of our simulation model for the Agents in Column search pattern.

Due to the nature of the deterministic models, we potentially have areas of significantly better coverage and areas of relatively little coverage. Spacing between detection locations cause these disparities in coverage. This can be seen in Figure 5.1 where spacing between agents can leave gaps in sweeps and discretization of time can leave gaps between glimpses. Furthermore, the presented framework is not explicitly designed to optimize the searchers' search pattern; therefore, a limitation of the current implementation is that some of the agents will search outside of the search area at the end of each sweep if the length of the search area is not an exact multiple of the number of searchers and the spacing between them.

5.1.2 Random Walk Search Pattern

Our Random Walk statistical model appears to be less effective in predicting the expected time to first detection compared to the deterministic search models, but still is demonstrated to be a useful model. The number of searchers, the searchers' velocity, and the target's velocity are seen to be the dominant factors in the statistical prediction model, which is consistent with the intuition for what factors are significant in overall search performance as measured by the time to first detection. We expect additional searchers to benefit the search performance, since increasing the number of searchers should increase the rate of coverage during the search, which leads to the searchers finding the target faster. This relationship corresponds to the negative coefficient for the number of searchers in the prediction equation (refer to Equation 4.7). Furthermore, an increased searcher velocity allows faster coverage of the search area. Due to the random movement of the Random Walk search pattern, searchers do not follow a predetermined route, preventing them from searching the exact same location and providing a more consistent coverage of the search area, unlike in the Agents Abreast and Agents in Column search patterns. Additionally, every agent will turn around when it gets to the edge of the search area concentrating their efforts inside of the search area.

The velocity of the target plays a large role as well, but the nature of its impact on the model is harder to ascertain. From fundamentals of search theory, one might expect that the movement of the target to effectively enhance the speed of the searchers during random search, contributing to a faster time to first detection referred to as the dynamic enhancement effect of target motion [6]. However, as the coefficient in the prediction model governing

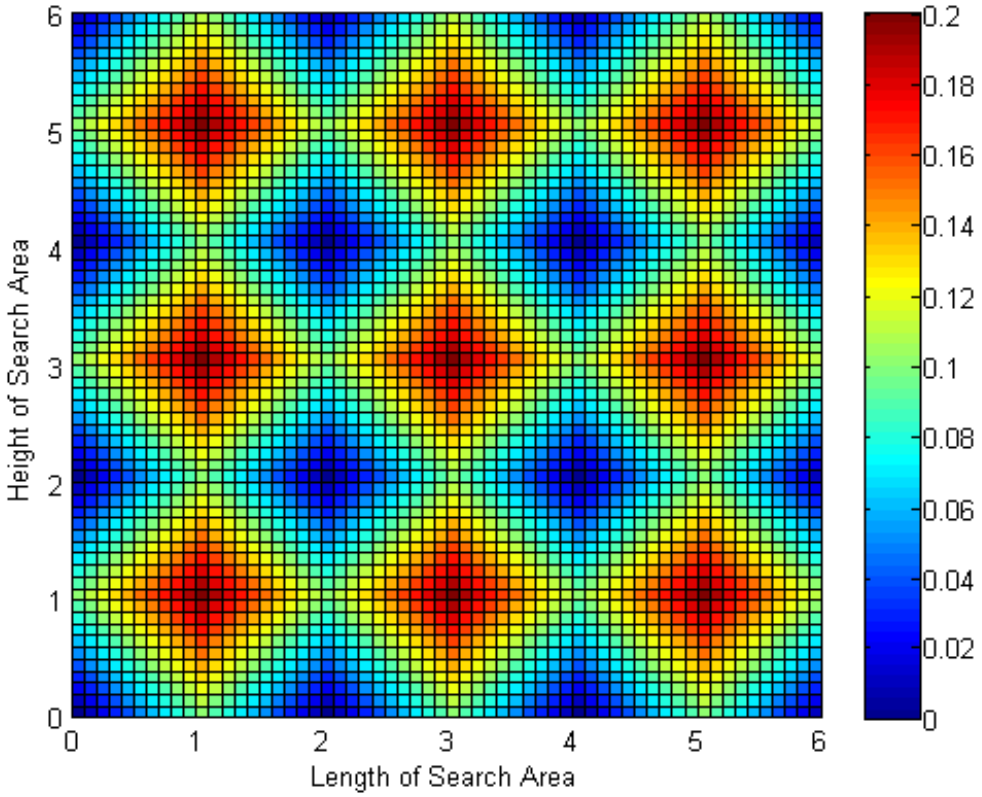


Figure 5.1: Gaps in coverage caused by excessive spacing and velocity in deterministic searches. This is caused by detections occurring once per timestep and the difference in location of the searcher at the beginning of each timestep. Red indicates a higher chance for detection while blue indicates a lower chance for detection or gaps.

the impact of target velocity is positive, the generated model predicts that, in practice, the opposite effect is exhibited. We believe that the restriction on the turning rate of the target causes this effect. Since the target's motion is not completely random, as its change of direction is limited to a certain bearing off its current heading, its movement is biased towards the direction it is currently facing, inducing it to move towards one of the boundaries of the search area. This leads us to conclude that the faster the target moves, the faster it will reach the edge of the search area, reducing the time inside the search area where it can be detected. Considering that if the target is restricted to the search area, the target's velocity would solely contribute to the effective velocity of the searcher rather than influence its

ability to leave the search area as well. The exact nature of this relationship merits further investigation in future work.

Of the three statistical models examined in this thesis, the Random Walk model performed the worst in predicting expected times to first detection for the optimal case.

5.1.3 Model Comparison

Of the three investigated search models, the Agents Abreast and Random Walk methods seemed to gain the greatest advantage from utilizing a swarm. While the Agents Abreast and Random Walk search patterns have comparable probabilities of detecting the target and expected time to first detection, their search performance dominates the Agents in Column model in both areas. We believe that there are several reasons for these results. When assessing the effectiveness of the Agents in Column search pattern, we notice that the rate of coverage does not increase with the number of agents; rather, the impact is to increase the probability of detection due to the multiple detection attempts of the target on each pass. As such, the searchers will most likely find the target after a single pass, and thus, the expected time to first detection should be a little over one half of the time it takes to complete one coverage sweep of the search area (for uniform prior probability of the initial target location), which we see reflected in the generated results. Furthermore, the target is also moving, making it possible for it to evade a pass by the column of searchers. These considerations combine to make the performance of the Agents in Column search pattern bimodal, either passing close enough to the target to detect it on the first pass, or completely missing the target for the entire replication.

Conversely, the Agents Abreast and Random Walk search patterns distribute their search efforts over a much wider area. This means that the probability of the searchers finding the target on the first pass may be smaller, but they cover the search area significantly faster. As we utilize homogeneous searchers for both the Agents Abreast and Random Walk search patterns, the time to effectively complete a sweep of the search area is inversely proportional to the number of searchers as seen in Equations 5.1 and 5.2, whereas the time for the Agents in Column pattern to complete a sweep of the search area remains the same as that for a single agent. This allows the Agents Abreast and Random Walk search patterns to cover nearly the entire search area enough times to reach the point of diminishing re-

turns significantly faster than the Agents in Column search pattern. We consider this to be the primary reason for the significantly slower time to first detection in the Agents in Column search pattern. It is worth noting that changes to the simulation implementation of all three search patterns could greatly enhance the fidelity of the simulation and potentially reveal improved search performance. For example, we could partition the search area and break the column of searchers into multiple columns, each column performing a lawnmower search over the partition. In this manner we could significantly reduce the effect of diminishing returns we see in the current simulation model implementation.

$$
\begin{aligned}
F_{\mathrm{AA}}(t) &= \sum_{i=1}^{M} \gamma_i t \\
F_{\mathrm{AA}}(t) &= M\gamma_i t \\
\implies \quad t^* &= \frac{F_{\mathrm{AA}}(t)}{M\gamma_i}
\end{aligned}
\tag{5.1}
$$

$$
\begin{aligned}
F_{\mathrm{RW}}(t) = 1 - e^{-\sum_{i=1}^{M} \gamma_i t} \quad &= \quad 1 - e^{-M\gamma_i t} \\
e^{-M\gamma_i t} &= 1 - F_{\mathrm{RW}}(t) \\
-M\gamma_i t &= \ln(1 - F_{\mathrm{RW}}(t)) \\
\implies \quad t^* &= \frac{\ln(1 - F_{\mathrm{RW}}(t))}{-M\gamma_i}
\end{aligned}
\tag{5.2}
$$

Another insight that merits further analysis is to quantify the upper decile of the timesteps before which 90 percent of the simulation trials have detected the target. For the Agents Abreast and Random Walk models, the upper decile occurs before 1500 timesteps for 75 percent of all design points, and 2200 timesteps represents the upper decile for all three swarm search patterns studied in this thesis, as seen in Figure 5.2. This preliminary analysis potentially provides additional insight, for example, that the search mission duration (i.e., simulation time) can be shortened substantially and can still lead to acceptable search performance. The slower effective search performance of the Agents in Column pattern is

also seen in the upper decile, being nearly five times longer than either of the other two models, as seen in Figure 5.2. Such further analysis may additionally yield operationally relevant insights of use to future employment of such swarm search capabilities.

5.2 Operational Applications

This framework provides the flexibility and robustness to apply to a wide range of operational areas. We include in this section some illustrative mission areas in the maritime environment that could leverage this framework to explore implementing swarm search systems for enhanced search effectiveness. Further investigations, perhaps using specific data associated with the mission area of interest, may identify further utility of such swarm search approaches.

5.2.1 Search and Rescue

The United States Coast Guard (USCG) responds to tens of thousands of distress calls a year, saving thousands of lives. Nearly twenty percent of those in distress are still lost [40]. The capabilities of unmanned systems can greatly increase the number of searchers the USCG may be able to employ in their search and rescue missions. As such, it is important for the USCG to test the effectiveness of their search strategies prior to implementing them in operational contexts where lives are at stake. The framework developed and proposed in this thesis aims to provide the flexibility necessary to explore a wider variety of search strategies potentially relevant to the USCG.

The major components for the USCG search model would likely include heterogeneous search platforms, sensor models including visual detection models, target characteristics and movement including drift conditions, and additional search agent characteristics. The need for heterogeneous search agents arises from the variety of search and rescue units (SRUs) that the USCG currently utilizes for their searches, such as National Security Cutters, rotary- and fixed-wing manned aircraft, and imminently UXS as well. Furthermore, the need may arise for multiple target types in the same model, as there may be personnel in the water, as well as vessels in distress. Since the USCG would be utilizing a variety of sensors, including the human eye in addition to, e.g., unmanned target recognition, different sensor characteristics would also need to be used. Ocean currents and weather conditions would impact both the target's movement, and future candidate sensor capabilities could

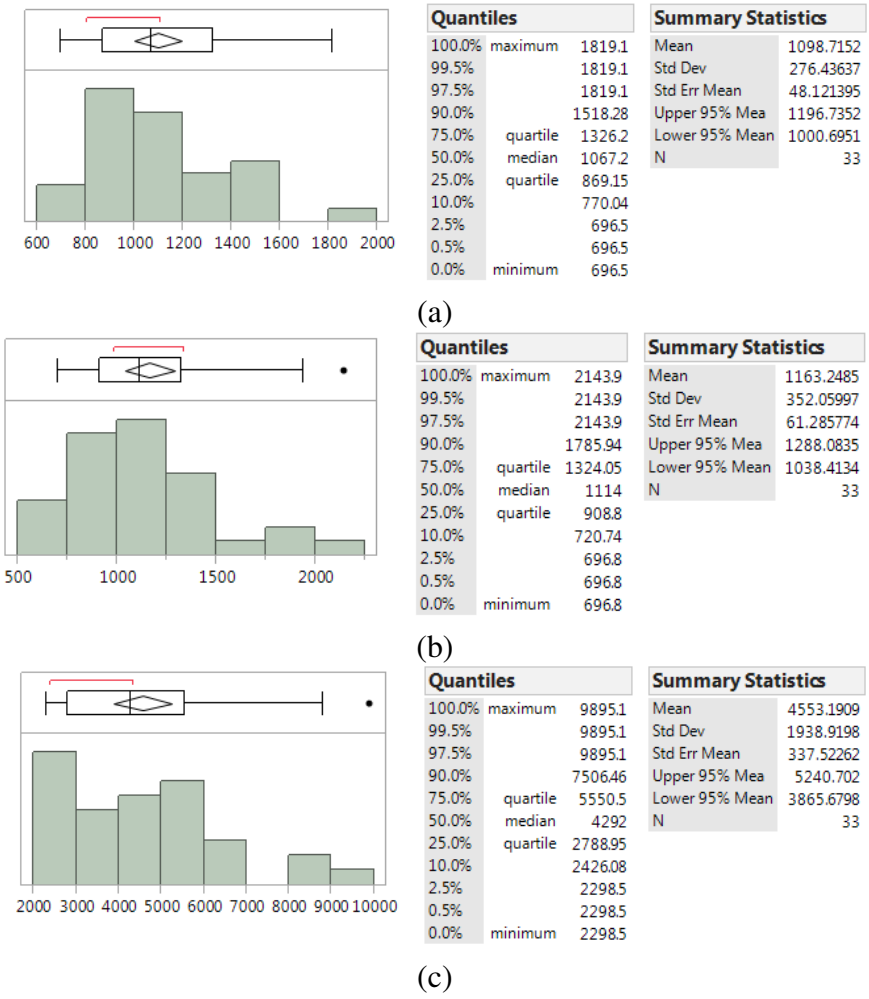


Figure 5.2: Distribution of Upper Decile for the expected time to first detection. Agents Abreast (a) and Random Walk (b) perform comparably; however, Agents in Column (c) perform significantly worse than either the Agents Abreast or Random Walk patterns.

also be incorporated into the respective modules in the proposed simulation and analysis framework. The complexity of the situation requires a complex model to fully capture the search efforts. However, the modular nature of the proposed framework allows straightforward integration of the above components into a single swarm search model and analysis methodology.

5.2.2 Mine Warfare

Mine warfare poses a significant threat to any maritime force as mines are relatively cheap and easy to employ. Furthermore, locating, identifying, and neutralizing mines is a time- and resource-intensive, let alone dangerous, task. Recent advances in unmanned systems have led to the development of several unmanned underwater mine hunting systems [41], [42]. Furthermore, surface ships often integrate with mine hunting efforts, providing different capabilities than their underwater counterparts. The mine threat exists in many different situations, requiring a flexible tool to determine the best method to employ resources for the situation.

A common tactic in mine warfare is the saturation of an area with a large number of mines, i.e., a mine field, in an attempt to deny that area to the enemy. When faced with such situations, U.S. and allied forces must have the ability to clear routes through the mined area to enable access by friendly assets. Such an operational setting can leverage the proposed framework, in which the employment of a large number of mine-sweeping and clearance assets, potentially unmanned, can be evaluated. In this case, it is possible to populate the environment with a large number of targets, as well as decoys if desired. By utilizing a module to employ a search pattern focused around certain routes through the minefield, we could determine the effectiveness of clearing that area. By implementing false alarms, or decoys, we can determine the amount of time wasted on identifying non-mine objects. By implementing different search agent and target characteristics, we could also model differences between searchers, such as underwater systems and surface systems, as well as different types of mines, such as bottom mines and moored mines, respectively.

5.2.3 Submarine Warfare

Submarine warfare is particularly well suited to search problems, given that the submarine is inherently a covert, nominally intelligent, adversary operating in a large and relatively unrestricted environment. Searching for submarines occurs in a variety of different situations. In scenarios envisioning contested maritime waters, often times adversary submarines may be employed to act as deterrents or scouts. Alternatively, when transiting and escorting a friendly High Value Unit (HVU), it is essential to clear the area around the HVU, necessitating the hunt for enemy undersea threats. Moreover, it may also be necessary to monitor the transit of enemy submarines through choke-points, either to know that they have de-

ployed or to initiate further tracking methods. Given the diversity of undersea warfare scenarios where submarines are employed, anti-submarine warfare and effective search for enemy assets may also potentially benefit from this thesis work.

As motivation for this thesis, Chapter 2 proposed an anti-submarine warfare scenario requiring area clearance of enemy submarine threats. As also mentioned, escorting a HVU is of high importance, and such a scenario could also be addressed using the framework presented in this thesis by changing the searchers' pattern to clear an area around the HVU before setting up a picket around the area of interest, with additional searchers clearing a larger area in advance of the transiting HVU. Furthermore, the target's behavior could be modified to emulate their approach of the HVU to simulate a threat. Additionally, a more appropriate measure of performance that could be captured by the modular simulation framework would be to measure, for example, the ability of the target to approach within a set distance from the HVU within a set time period.

5.3 Future Work

While the representative demonstration of the proposed simulation and analysis framework in this thesis highlight its contribution for assessing swarm search models, there are many avenues of future research and extensions for further improvement. Further research and modifications could greatly expand the relevance and number of applicable situations, as well as improve the interface between the software implementation and the operations analyst.

5.3.1 Target Distribution

In the situations modeled in this thesis, we generated the target's initial location as drawn from a uniform distribution over the search area, but availability of intelligence on target location may inform a different distribution. We are interested in studying the impact of different target distributions on the search effectiveness. In the context of the modular framework, this addition would be easy to implement by restructuring the `initial target position` module by incorporating a probabilistic bias to the starting location. For example, we could implement scenarios where we have time late information on the target, so we know the maximum distance the target could be from a given datum, and further

knowledge or assumptions about the target could further refine the distribution of target locations.

We could also use the `initial target position` module to explore how adversarial tactics affect current and future search capabilities. If we believe that the targets arrange themselves in a specific formation or along certain geological features, such as staying close to the coastline, we can determine how this affects the performance of our searches. A random search or one focused on uniform coverage of the entire search area may not be as effective as one that places a majority of its efforts in the areas we suspect the target to be located. Analyzing how the starting location of the target affects different search techniques may allow for effective modification of search tactics.

Furthermore, such future work can enable a better measure of the quality of information and target location intelligence. By comparing our measures of effectiveness with the uniform target distribution to those where we have a biased starting location, we may determine how much the additional intelligence improves our search. This comparative study can allow operational planners to calculate the amount of resources to be allocated to gaining that intelligence versus putting them into other areas of the search. It may also be possible to determine this target distribution from previous searches. Due to the limited endurance of most unmanned systems, we could utilize this knowledge on how target distributions affect search performance to conduct initial coarse searches to allow follow-on, higher-resolution searches to perform more effectively.

Further analysis into how different initial target distributions impacts our search will help further develop the robustness of the proposed framework and methodology.

5.3.2 Search Patterns

For the presented research, we focused on three relatively basic search patterns. Two of these patterns are deterministic, based on an ideal lawnmower search pattern, and the third is a generic random search. While these provide an adequate baseline and are useful for showing the applicability of the proposed framework, the true purpose of the framework is exploring search strategies that take full advantage of swarm behaviors. As such, research into different search patterns would provide insight into the advantages and disadvantages that each provides.

As outlined in Chapter 3, the search pattern is controlled by the `searcher movement` module. This decomposition allows considerable flexibility in the search patterns that we can explore, and enables the development of dynamic search patterns which change based on changing circumstances in the search evolution. For example, we could utilize the framework to investigate the differences between a klinokinetic and an orthokinetic search process [43]. For a klinokinetic search process, a searcher's turning rate increases as it finds itself closer to its desired zone. This change induces the searcher's path to become more complex, causing it to remain in the general vicinity. Conversely, for the orthokinetic search process, the searcher's forward speed decreases as it finds itself closer to its desired zone, which causes the searcher to linger in the vicinity. By linking these behaviors to target detections, we can potentially implement an effective tracking system. Furthermore, if the agents act independently, without communication between themselves or a central server, we could use a snap shot of the searchers' locations at a specific time, and ascertain the target's location through the concentration of searchers. These types of random search behaviors are often found in nature and their application could be investigated in regards to a swarm search model.

Similarly, the framework makes a large amount of data available to all of the modules, which would provide the basis for exploring other nature-inspired search strategies. For example, this framework could be utilized to simulate ant-based foraging models, where each agent deposits digital pheromones which guide the actions of the other agents. In this way, we could investigate the feasibility of integrating this type of search strategy into swarms of unmanned systems.

Numerous search strategies are currently being explored that could influence the future of swarm searches. These strategies offer their own advantages and disadvantages and have yet to be fully explored in an operationally relevant environment or context. The framework developed herein provides a foundation which allows the exploration of these search strategies in a multitude of scenarios.

5.3.3 Sensor Characteristics

The consideration of different sensor characteristics provides a wide area of future development as well. As sensors and their detection characteristics define a significant component

in search models and search performance, analyzing the effectiveness of sensors is vital to informing both design decisions as well as search asset employment strategies.

Often times, sensor performance changes depending on the conditions that it is used in. For example, some sensors work better at certain speeds, or their ability to correctly identify a target decreases as the speed of the search platform utilizing it increases. Other times, environmental factors play a major part in the sensor capabilities, such as heavy fog decreasing effective ranges of visual detectors. It is important to capture these characteristics, especially in situations where searcher behaviors or environmental conditions may be changing.

Our framework could be extended in a straightforward manner to address many of these considerations. The ability to easily change the `sensor` modules allows for the rapid testing of multiple types of sensors incorporated into a number of different search strategies. Furthermore, characteristics of the individual agents, such as their search speeds, as well as information about the environment, is already passed to each module in the current implementation of the software, allowing the sensor's detection curve to potentially adapt to current operating conditions [19].

Furthermore, real-world sensors are never perfect. The sensor profile used in the demonstration of the framework was assumed to have a limited detection capability, involving false negative detections, even at its most effective range. Further, due to noise in the sensor and the environment, there exists the possibility of false alarms. While false alarms do not directly impact our ability to detect the real target, it does create confusion and complicates the search [19]. Such distractions impact search efforts considerably, and as such, future search models can incorporate an inspection behavior that requests additional search resources upon a possible contact. Moreover, with the introduction of false alarms into a given search model, we are able to explore the effectiveness of search strategies that are adaptive (versus nonadaptive) to (both true and false) positive detections.

Implementation of different sensors into the proposed framework involves changing the `detection` module. For the different sensor detection curves or those that depend on dynamic agent characteristics, this modification would primarily involve the sub-function that determines the probability of detection to reflect, e.g., a different sensor detection curve. For the introduction of false alarms, a more substantial development effort would be

necessary, as an additional variable which triggers follow-on behaviors for search agents (such as if an actual detection was made) would need to be implemented.

5.3.4 Levels of Communication

The level of communication between agents is another important aspect of future concepts for swarm search. In an ideal situation, each agent would have the same knowledge as all other agents in real time. Due to limitations in technology and operational constraints, providing enough bandwidth for many systems to communicate necessitates looking for other communication options besides a complete network [11]. Traditionally, a hub-and-spoke model has been relied on for command and control of multiple agents, where a central server communicates with all agents; however, several other potential solutions exist and should be explored as well.

A central server provides a hub for inter-agent communication to pass through before it branches out to agents. The hub-and-spoke model provides many benefits, including simplicity, for a model containing n agents only needs $n-1$ spokes to connect them. Furthermore, the simplicity of the network can potentially lead to an efficient use of bandwidth between agents. The main drawback of this network configuration is the lack of robustness, for the loss of the central server would destroy the network.

The most comprehensive communication scheme is a complete network where each agent passes information to every other agent. This topology provides each agent with the most up-to-date information while maintaining the most reliable flow of information; however, such a complete network also means that as the number of agents grows, the bandwidth required grows exponentially and quickly become infeasible. One candidate approach could involve establishing communication links to only a set of closest neighbors of each agent, which could significantly reduce the number of communication paths. This method makes it possible to potentially free considerable amounts of resources, although also adds several areas of complexity, such as now possibly needing to address dynamic networks where an agent's closest neighbors may change with time. Furthermore, depending on the size of the network, and the number of closest agents within communication range, several disconnected sub-networks may be formed with no communication between them.

A unique approach relies on indirect communication between agents. By utilizing the onboard sensors of the agent to read aspects of their surroundings, it is possible to relay information and influence the behavior of the agents. In some approaches in the multi-agent literature, the primary method of indirect communication relies on the relative location and velocity vector of nearby agents, but if it is possible to expand this to different markers, then it could potentially be possible to mimic behaviors in nature such as the pheromone-laying performed by ants without the use of communication bandwidth to transmit such digital pheromones [15].

5.3.5 Reactive and Adversarial Targets

Assuming that the target is ignorant of search attempts against it is a rather naïve assumption. In most cases, the target will be able to detect the searchers and respond to their search attempts. In other cases, the act of the searcher attempting to sense the target will alert the target, such as in the case of active radar or active sonar. As such, the target's movement pattern and strategy also plays a large role in the search process.

Currently, the `target movement` module does not incorporate additional information in the rest of the search evolution. By linking target behaviors, such as moving away from the nearest searcher, to certain conditions, such as the searchers being within detection range, future study could explore how these actions affect the search performance. Furthermore, introduction of additional modules that incorporate a separate detection range by the target in order to determine if the target senses the searchers could enable responses to counter detections.

We can also integrate behaviors into our searchers, programming them to react to the target. Often times search patterns change when a target is detected. Such adaptive behaviors include attempts to localize and close on the target in order to perform detailed target identification. In other cases, cooperating search assets may attempt to "herd" the target into or out of an area of interest. By adding these behaviors to the simulation model, these increasingly complex behaviors and their impact on the overall search strategy could also be assessed, to include identification of the key factors impacting the searchers' ability to perform these adaptive search behaviors.

5.3.6 JMP Integration

Recent changes to the statistical software, JMP, provides a wealth of new tools to the already powerful suite of features [44]. These features include the ability to open and close connections between JMP and MATLAB directly for exchange of data and submission of code for execution. This feature allows the potential for significant integration between the statistical analysis tool and the underlying foundation of the simulation framework presented in this thesis. Implementing this cross-application integration could greatly simplify the development and analytic process of new search models.

Additionally, JMP possesses an application development kit, which could enable the creation of self-contained, standalone applications in JMP, which can then be deployed at other stations. Combining this feature with the ability to construct a comprehensive Graphical User Interface (GUI) may potentially allow the development of a tactical support and/or analysis tool. Such a decision aid could assist the analyst or system engineer in developing, implementing, and analyzing different swarm search models in various contexts. Furthermore, such software could also allow the production of user-friendly trade-space graphics to help users make decisions, potentially even with a limited understanding of the underlying model or program. As such, the ability to integrate the MATLAB-based modeling capability of our framework with the statistical analysis tools provided by JMP could be a useful avenue for future development efforts.

List of References

[1] M. Eaglen and J. Rodeback, "Submarine arms race in the Pacific: The Chinese challenge to US undersea supremacy," Heritage Foundation, Washington, DC, Tech. Rep., Feb. 2010.

[2] O. R. Cote, Jr., *The Third Battle: Innovation in the U.S. Navy's Silent Cold War Struggle with Soviet Submarines*, (Newport Paper Number 16). Newport, Rhode Island: NWC, 2003, pp. 3-116.

[3] S. J. A. Edwards, *Swarming on the Battlefield*. Santa Monica, CA: Rand, 2000.

[4] R. O. Work and S. Brimley, "Preparing for war in the robotic age," Center for a New American Security, Washington, DC, Tech. Rep., 2014.

[5] J. A. Sauter, R. S. Mathews, K. Neuharth, J. S. Robinson, J. Moody, and S. Riddle, "Demonstration of swarming control of unmanned ground and air systems in surveillance and infrastructure protection," in *IEEE Conference on Technologies for Homeland Security*, Boston, MA, 2009, pp. 51–58.

[6] A. R. Washburn, *Search and Detection*, 4th ed. Institute for Operations Research and the Management Sciences, 2002.

[7] J. J. Corner, "Swarming reconnaissance using unmanned aerial vehicles in a parallel discrete event simulation," M.S. Thesis, Air Force Institute of Technology Wright Patterson AFB OH School of Engineering and Management, 2004.

[8] J. Sauter and R. Matthews, "Performance of digital pheromones for swarming vehicle control," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, Utrecht, Netherlands, 2005, pp. 903–910.

[9] M. A. Kovacina, D. Palmer, G. Yang, and R. Vaidyanathan, "Multi-agent control algorithms for chemical cloud detection and mapping using unmanned air vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002, pp. 2782–2788.

[10] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Chichester, West Sussex: John Wiley & Sons, 2005.

[11] P. Dasgupta, "A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 3, pp. 549–563, 2008.

[12] D. J. Nowak, I. Price, and G. B. Lamont, "Self organized UAV swarm planning optimization for search and destroy using swarmfare simulation," in *Winter Simulation Conference*, Dayton, OH, 2007, pp. 1315–1323.

[13] A. Banks and J. Vincent, "Exploring the performance of natural search strategies for the control of unmanned autonomous vehicles," *Journal of Navigation*, vol. 62, no. 02, pp. 283–301, Mar. 2009.

[14] L. Barnes, M. M.-A. Fields, and K. Valavanis, "Unmanned ground vehicle swarm formation control using potential fields," in *Mediterranean Conference on Control Automation*, Athens, Greece, June 2007, pp. 1–8.

[15] M. Flint, M. Polycarpou, and E. Fernandez-Gaucherand, "Cooperative control for multiple autonomous UAV's searching for targets," in *41st IEEE Conference on Decision and Control*, Las Vegas, NV, 2002, pp. 2823–2828.

[16] R. R. Pitre, X. R. Li, and D. R. DelBalzo, "UAV route planning for joint search and track missions An information-value approach," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 3, pp. 2551–2565, 2012.

[17] D. J. Pack and G. W. York, "Developing a control architecture for multiple unmanned aerial vehicles to search and localize RF time-varying mobile targets: Part I," in *IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp. 3954–3959.

[18] "Course Introduction," class notes for Search Theory and Detection, Dept. of Operations Research, Naval Postgraduate School, Monterey, CA, Spring 2013.

[19] D. H. Wagner, W. C. Mylander, and T. J. Sanders, *Naval Operations Analysis*, 3rd ed. Annapolis, MD: Naval Institue Press, 1999.

[20] B. Liu, R. Zhang, and C. Shi, "Analysis of foraging behavior in ant swarms based on Starlogo simulation," in *IEEE International Conference on Robotics and Biomimetics*, Sanya, China, 2007, pp. 810–815.

[21] H. V. D. Parunak, S. Brueckner, and J. Sauter, "Digital pheromone mechanisms for coordination of unmanned vehicles," in *International Joint Conference on Autonomous Agents and Multiagent Systems*. Bologna, Italy: ACM, 2002, pp. 449–450.

[22] J. E. Kline, "Red Dragon at Sea Scenario 2025," unpublished.

[23] I. Easton, "China's evolving reconnaissance- strike capabilities: Implications for the U.S.-Japan alliance," The Japan Institute of International Affairs: Project 2049 Institute, Chiyodaku, Tokyo, Japan, Tech. Rep. February, 2014.

[24] D. Cheng, "Meeting the challenge of Chinese expansionism on the East Asian littoral," The Heritage Foundation, Washington, DC, Tech. Rep. 4135, 2014.

[25] J. L. Karotkin, "Trend's in China's Naval Modernization," Washington, DC, p. 12, 2014.

[26] L. Fuell, "Broad Trends in Chinese Air Force and Missle Modernization," Washington, DC, p. 11, 2014.

[27] C. Murray and K. Hsu, "China's new fishing regulations seek to justify and consolidate control in the South China Sea," U.S.-China Economic and Security Review Commission, Washington, DC, Tech. Rep., 2014.

[28] T. L. Grund, Jr., "Distributed Air Wing," 2013.

[29] A. M. Law, *Simulation Modeling & Analysis*, 4th ed., K. E. Case and P. M. Wolfe, Eds. New York, NY: McGraw-Hill, 2007.

[30] O. A. Yakimenko, *Engineering Computations and Modeling in MATLAB/Simulink*. American Institute of Aeronautics and Astronautics, 2011.

[31] MathWorks. (2014, Jun. 25). Strategies for efficient use of memory - MATLAB & Simulink. [Online]. Available: http://www.mathworks.com/help/matlab/matlab_prog/strategies-for-efficient-use-of-memory.html

[32] J. P. C. Kleijnen, S. M. Sanchez, T. W. Lucas, and T. M. Cioppa, "A users guide to the brave new world of designing simulation experiments," *INFORMS Journal on Computing*, vol. 17, no. 3, pp. 263–289, Aug. 2005.

[33] S. M. Sanchez, T. W. Lucas, P. J. Sanchez, C. J. Nannini, and H. Wan, "Designs for Large-Scale Simulation Experiments, with Applications to Defense and Homeland Security," in *Design and Analysis of Experiments, Special Designs and Applications*, K. Hinkelmann and O. Kempthorne, Eds. Hoboken, New Jersey: John Wiley & Sons, 2012, vol. 3, no. 1986, ch. 12, pp. 1–25.

[34] S. Sanchez and P. Sanchez, "Very large fractional factorial and central composite designs," *ACM Transactions on Modeling and Computer Simulation*, vol. 15, no. 4, pp. 362–377, 2005.

[35] S. Sanchez and H. Wan, "Work smarter, not harder: A tutorial on designing and conducting simulation experiments," in *Winter Simulation Conference*, Berlin, Germany, 2012, pp. 47–57.

[36] H. Vieira, S. Sanchez, K. H. Kienitz, and M. C. N. Belderrain, "Improved efficient, nearly orthogonal, nearly balanced mixed designs," in *Winter Simulation Conference*. Phoenix, AZ: IEEE, 2011, pp. 3605–3616.

[37] S. M. Sanchez and T. W. Lucas. (2014, Jul. 11). SEED Center for data farming. [Online]. Available: http://harvest.nps.edu/

[38] J. L. Devore, *Probability & Statistics for Engineering and the Sciences*, 8th ed., M. Julet, Ed. Boston, MA: Brooks/Cole, 2012.

[39] D. C. Montgomery, E. A. Peck, and G. G. Fining, *Introduction to Linear Regression Analysis*, 5th ed., P. Bloomfield, Ed. Hoboken, New Jersey: John Wiley & Sons, 2012.

[40] United States Coast Guard, "Coast Guard search and rescue statistics , 1964-2001," U.S. Department of Homeland Security, Washington, DC, Tech. Rep., 2001.

[41] D. Parsons, "Bigger brains, better batteries will enable new missions for robotic submarines," *National Defense*, vol. 98, no. 718, pp. 28–29, 2013.

[42] NSWC Panama City. (2014, Aug. 10). MK18 Kingfish UUV deployed to 5th Fleet, [Online]. Available: http://www.navy.mil/submit/display.asp?story_id=75039

[43] S. Kernbach, Ed., *Handbook of Collective Robotics: Fundamentals and Challenges*. Singapore: Pan Stanford, 2013.

[44] M. Proust, "New features in JMP 11," SAS Institute Inc., Cary, NC, Tech. Rep., 2013.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California